Artificial Intelligence and Logistics

IJCAI 2011 Workshop Proceedings

Kerstin Schill, Bernd Scholz-Reiter, Lutz Frommberger (Eds.)

SFB/TR 8 Report No. 028-07/2011
Report Series of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition
Universität Bremen / Universität Freiburg

Contact Address:

Dr. Thomas Barkowsky
SFB/TR 8
Universität Bremen
P.O.Box 330 440
28334 Bremen, Germany

Tel +49-421-218-64233
Fax +49-421-218-64239
barkowsky@sfbtr8.uni-bremen.de
www.sfbtr8.uni-bremen.de

PROCEEDINGS

2ND WORKSHOP ON

ARTIFICIAL INTELLIGENCE AND LOGISTICS

(AILOG-2011)

Kerstin Schill, Bernd Scholz-Reiter, Lutz Frommberger (eds.)

July 16, 2011

22nd International Joint Conference on Artificial Intelligence (IJCAI)

Barcelona, Spain

SFB/TR 8 SPATIAL COGNITION

SFB 637 Autonomous Logistics

2ND WORKSHOP ON ARTIFICIAL INTELLIGENCE AND LOGISTICS (AILOG-2011)

ORGANIZERS

Kerstin Schill
SFB/TR 8 "Spatial Cognition", University of Bremen, Germany
kschill@sfbtr8.uni-bremen.de

Bernd Scholz-Reiter
SFB 637 "Autonomous Cooperating Logistic Processes", University of Bremen, Germany
bsr@biba.uni-bremen.de

Lutz Frommberger
SFB/TR 8 "Spatial Cognition", University of Bremen, Germany
lutz@informatik.uni-bremen.de

PROGRAM COMMITTEE

Ana Bazzan (Universidade Federal do Rio Grande do Sul, Brazil)
John Bateman (University of Bremen, Germany)
Jürgen Branke (University of Warwick, UK)
Neil A. Duffie (University of Wisconsin-Madison, USA)
Boi Faltings (EPFL Lausanne, Switzerland)
Torsten Hildebrandt (University of Bremen, Germany)
Eyke Hüllermeier (University of Marburg, Germany)
Kap Hwan Kim (Pusan National University, Korea)
Stefan Kirn (University of Hohenheim, Germany)
Herbert Kopfer (University of Bremen, Germany)
Andreas D. Lattner (University of Frankfurt, Germany)
Martin Lauer (Karlsruhe Institute of Technology, Germany)
Ramon López de Màntaras (IIIA-CSIC, Spain)
Jacek Malec (Lund University, Sweden)
Norman Sadeh (Carnegie Mellon University, USA)
Hedda Schmidtke (Karlsruhe Institute of Technology, Germany)
Jaime Sichman (Universidade de So Paulo, Brazil)
Gerhard Weiss (Maastricht University, Netherlands)
Katja Windt (Jacobs University Bremen, Germany)
Stefan Wölfl (University of Freiburg, Germany)

ORGANIZING INSTITUTIONS

SFB/TR 8 "Spatial Cognition"
University of Bremen and University of Freiburg, Germany
http://www.sfbtr8.spatial-cognition.de

SFB 637 "Autonomous Cooperating Logistic Processes"
University of Bremen, Germany
http://www.sfb637.uni-bremen.de

TABLE OF CONTENTS

PREFACE

Global economy seems to have overcome the depression of the financial crisis in 2009. Markets are growing again, causing an increase in global material flows. Gaining and maintaining competitive advantages in such an environment is only possible if systems and processes to handle these material and associated information flows are properly designed and can fully utilize available, scarce resources.

Within this development, methods of Artificial Intelligence gain importance and can offer Logistics researchers new ways to face the problems of today's logistic systems. Vice versa, logistics provides a challenging area to apply and improve AI research. Engineering and operating such highly dynamic and complex systems should be a joint research effort for both, the Artificial Intelligence community and Logistics research.

From the increasing complexity of modern logistic processes emerges a great variety of AI methods employed. For example, ontologies are used to conceptualize complex systems in a formal way. Unforeseen circumstances require adaptive solutions. Neural networks and genetic algorithms are frequently used, and machine learning techniques of all flavors lead to flexible approaches tailored to specific demands. As processes become more and more distributed, especially multi-agent concepts become increasingly important. Questions of coordination and cooperation have to be tackled, also leading to new developments in formal approaches such as distributed planning, distributed constraint optimization problems, or qualitative spatial reasoning. Increasingly, the domain of mobile robotics becomes evident, and concepts developed for autonomous robots can also be applied successfully to many tasks in Logistics.

The AILog workshops aim at aggregating this variety of methods and applications. Being located at the major international AI conferences, we hope for an intense contact between experts in Logistics and experts in AI in order to trigger mutual exchange of ideas, formalisms, algorithms, and applications. While this has been successfully achieved with the 2010 AILog workshop, we hope for interesting and fruitful discussions from this year's workshop as well. From the submitted manuscripts we selected 11 papers for presentation at the workshop after a thorough peer-review process.

We want to thank all the authors for their contribution and the members of the program committee and the external reviewers for the substantial feedback they provided on the submitted manuscripts, and Torsten Hildebrandt for his significant help in the workshop organization. We thank collaborative research centers "SFB 637 Autonomous Cooperating Logistic Processes" and "SFB/TR 8 Spatial Cognition", funded by the German Research Foundation (DFG), for their support. Financial support provided by the University of Bremen is also gratefully acknowledged. Last but not least we thank the IJCAI 2011 organization, in particular Adele Howe and Carles Sierra, for helping to make AILog-2011 possible.

We are looking forward to an inspiring exchange of ideas and fruitful discussions in Barcelona.

Kerstin Schill
Bernd Scholz-Reiter
Lutz Frommberger

(AILog-2011 organizers)

INVITED TALK

ORGANIC TRAFFIC CONTROL

JÜRGEN BRANKE, UNIVERSITY OF WARWICK, UK

Recent trends in manufacturing lead to new challenges in logistics, as problems become much more complex and dynamic. Luckily, recent technological advances make it possible to tackle these new challenges by providing effective on-line control mechanisms. Organic computing is a recent paradigm aimed at building technological systems that have more organic, or life-like properties, including learning, self-adaptation and self-configuration. After some general discussion of logistics and organic computing, the talk will present Organic Traffic Control as one exemplary applicaton of Organic Computing technology. The proposed approach continuously learns new and better traffic light control strategies in a stochastic and dynamically changing environment, and switches between them as appropriate.

TECHNICAL PAPERS

Challenges in Maintaining Minimal, Decomposable Disjunctive Temporal Problems

James C. Boerkoel Jr. and Edmund H. Durfee
Computer Science and Engineering, University of Michigan
Ann Arbor, MI
{boerkoel,durfee}@umich.edu

Abstract

In many scheduling applications, new scheduling constraints can arise dynamically due to exogenously determined events and preferences. In such environments, maintaining a set of all possible remaining schedules can be much more robust than selecting a single schedule. In this paper we discuss two properties, minimality and decomposability, that are necessary for faithfully representing the set of all remaining solutions to a Disjunctive Temporal Problem (DTP). We prove minimal and decomposable representations always exist for a consistent DTP. We also introduce metrics for comparing different minimal, decomposable DTP representations in terms of space and time efficiency and propose ideas for improving efficiency based on work from dispatching disjunctive schedules.

1 Introduction

In many scheduling applications, the actual duration of a specific activity, such as the transportation time between two locations, may be either uncertain, exogenously determined, or subject to unexpressed preferences. As an example, consider a truck that starts out at a depot and must make deliveries to three different locations by a predetermined deadline. While the truck can visit the three locations in any order, each order may have different implications on travel and processing time due to traffic congestion and the overhead involved in reshuffling inventory on the truck. In such applications, calculating a single plan with a specific schedule may be brittle to the dynamics and preferences involved in the problem. A more robust approach for dealing with uncertainty, dynamism, and unexpressed preferences in logistics and scheduling applications is to instead calculate the set of *all* feasible schedules. This approach efficiently supports queries of the form "When can I perform delivery A?", or "Can I make delivery X before I make delivery Y?".

However, the set of all feasible schedules generally grows exponentially in size as the number of events increases. Fortunately, there exist constraint-based problem formulations that are capable of *compactly* representing sets of feasible schedules [Dechter *et al.*, 1991; Stergiou and Koubarakis, 2000; Shah and Williams, 2008]. The most general of these is called the Disjunctive Temporal Problem (DTP), whose generality is required for representing the example problem described above. Faithfully representing the set of *all feasible* solutions requires establishing properties called *minimality* and *decomposability*. Minimality ensures that the complete set of solutions is represented while decomposability ensures that any consistent, partial schedule that respects constraints can be soundly extended to a complete solution schedule.

In this paper, we will review three important constraint-based scheduling problem formulations, including the Disjunctive Temporal Problem (DTP) in Section 2. We will demonstrate that the concepts of minimality and decomposability, which are well defined for some problem formulations, extend naturally to the more general DTP. We will also prove in Section 3 that despite fundamental differences from their less-complex predecessors, minimal, decomposable representations for consistent DTPs always exist. In Section 4 we explore the challenges associated with establishing minimal, decomposable DTP representations that are more efficient in terms of space (compactness) and the speed of reestablishing minimality and decomposability after an update. Gaining inspiration from important related work in dispatching disjunctive scheduling, we will identify specific challenges to, and propose ideas for, more efficient maintenance. Finally, we will conclude with some discussion and future work in Section 5.

2 Background

In this section we provide definitions necessary for understanding our contributions.

2.1 Simple Temporal Problem

We begin by adapting our description of the Simple Temporal Problem (STP) [Boerkoel and Durfee, 2011]. As defined by Dechter *et al.* [1991], the Simple Temporal Problem (STP), $\mathcal{S} = \langle V, C_{STP} \rangle$, consists of a set of timepoint variables, V, and a set of temporal difference constraints, C_{STP}. Each timepoint variable represents an event, and has an implicit, continuous numeric domain. Each temporal difference constraint c_{ij} is of the form $v_j - v_i \leq b_{ij}$, where v_i and v_j are distinct timepoints, and $b_{ij} \in \mathbb{R}$ is a real number bound on the difference between v_j and v_i.

To exploit extant graphical algorithms and efficiently reason over the STP *constraint network*, each STP is associated with a weighted, directed graph, $\mathcal{G} = \langle V, E \rangle$, called a *distance graph*.

The set of vertices V is as defined before (each timepoint variable acts as a vertex in the distance graph) and E is a set of directed edges, where, for each constraint c_{ij} of the form $v_j - v_i \leq b_{ij}$, we construct a directed edge, e_{ij} from v_i to v_j with an initial weight $w_{ij} = b_{ij}$. As a graphical short-hand, each edge from v_i to v_j is assumed to be bi-directional, compactly capturing both edge weights with a single label, $[-w_{ji}, w_{ij}]$, where $v_j - v_i \in [-w_{ji}, w_{ij}]$ and a weight w_{xy} is initialized to ∞ if there exists no corresponding constraint, $c_{xy} \in C_{STP}$. All times (e.g. 'clock' times) can be expressed relative to a special ***zero*** timepoint variable, $z \in V$, that represents the "start of time". Bounds on the difference between v_i and z can be thought of as "unary" constraints specified over a timepoint variable v_i. Moreover, w_{zi} and w_{iz} then represent the earliest and latest times, respectively, that can be assigned to v_i, and thus implicitly define v_i's domain. In this paper, we will assume that z is always included in V and that, during the construction of $\mathcal{G}$, an edge e_{zi} is added from z to every other timepoint variable $v_i \in V$. Examples of distance graphs that correspond to solutions for the example problem introduced in Section 1 (and formalized in Section 2.2) are in Figure 1. An STP is *consistent* if it contains at least one *solution*, which is an assignment of specific time values to timepoint variables that respects all constraints to form a *schedule*. Approaches for efficiently finding and representing the set of *all* solutions is presented in Section 2.4.

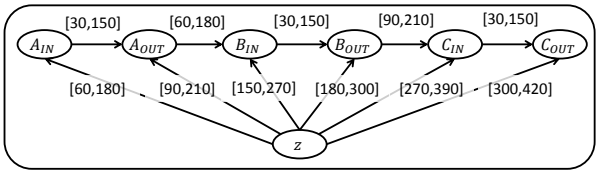
2.2 Disjunctive Temporal Problem

In order to represent a scheduling problem as an STP, the relative, partial order of all involved timepoints must be predetermined. However, many scheduling problems require making decisions over the relative order in which to execute activities. For example, capturing the fact that the truck can visit locations in *any* order requires *disjunctive* constraints. In comparison to the STP, the Disjunctive Temporal Problem (DTP), $\mathcal{D} = \langle V, C_{DTP} \rangle$, specifies a more general set of disjunctive constraints, C_{DTP}, where $c \in C_{DTP}$ represents a disjunction over a set of *any* temporal difference constraints. A constraint c takes the form $d^1 \vee d^2 \vee \cdots \vee d^k$ for some $k \geq 1$, where each disjunct d represents a typical temporal difference constraint over (possibly *different*) pairs of timepoints, $v_j - v_i \leq b_{ij}$. Note that an STP is a special case of a DTP where $k = 1$ for all constraints ($C_{STP} \subseteq C_{DTP}$ and subsequently, STP $\subseteq$ DTP).

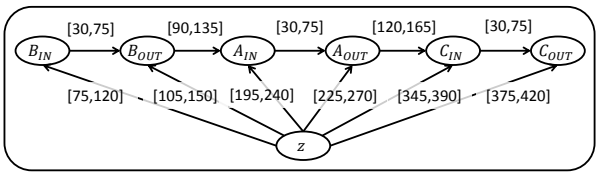
Equipped with the more general DTP representation, we now formally illustrate how to faithfully capture the constraints and other aspects of a more detailed version of the example logistics problem introduced in Section 1. We summarize this representation in Table 1. The problem involves a single truck that needs to make deliveries to three locations, A, B, and C. In addition to the zero timepoint z (where $z = 0$ represents the start time of the journey), there are timepoint variables representing the arrival X_{IN} and departure X_{OUT} of the truck to location X, for each of the three locations. The truck starts its day at a relatively centrally-located depot, and must make each delivery by various deadlines ($X_{OUT} - z \leq b_{DL(X)}$ $\forall X$ where $b_{DL(X)}$ is delivery X's deadline). Each location requires at least 30 minutes duration for unloading ($X_{IN} - X_{OUT} \leq -30$ $\forall X$). Note that constraints over transition

Trans. from Depot	Deadline	Min. Duration
$z - A_{IN} \leq -60;$	$A_{OUT} - z \leq 300;$	$A_{IN} - A_{OUT} \leq -30;$
$z - B_{IN} \leq -75;$	$B_{OUT} - z \leq 360;$	$B_{IN} - B_{OUT} \leq -30;$
$z - C_{IN} \leq -90;$	$C_{OUT} - z \leq 420;$	$C_{IN} - C_{OUT} \leq -30;$
Location to Location Transition (disjunctive)		
$A_{OUT} - B_{IN} \leq -60 \vee B_{OUT} - A_{IN} \leq -90;$ $B_{OUT} - C_{IN} \leq -90 \vee C_{OUT} - B_{IN} \leq -120;$ $A_{OUT} - C_{IN} \leq -120 \vee C_{OUT} - A_{IN} \leq -150$		

Table 1: Summary of the example logistics problem.



a) The minimal, decomposable STP distance graph corresponding to labeling: $\ell = \{A_{OUT} - B_{IN} \leq -60; B_{OUT} - C_{IN} \leq -90; A_{OUT} - C_{IN} \leq -120\}$



b) The minimal, decomposable STP distance graph corresponding to labeling: $\ell = \{B_{OUT} - A_{IN} \leq -90; B_{OUT} - C_{IN} \leq -90; A_{OUT} - C_{IN} \leq -120\}$

Figure 1: The distance graphs corresponding to minimal, decomposable solutions to the example problem (Table 1) corresponding to the three feasible labelings.

time (e.g., constraints of the form $X_{OUT} - Y_{IN} \leq b_{YX} \vee Y_{OUT} - X_{IN} \leq b_{XY}$ for some locations X and Y and bounds b_{YX} and b_{XY}), which include transportation time, are neither reflexive nor transitive due to traffic congestion and overhead involved in reshuffling inventory on the truck.

The DTP is often solved using a *meta-CSP* formulation, where each disjunctive temporal constraint $c \in C_{DTP}$ forms a *meta-variable* with a domain of *meta-values* composed of the set of possible disjuncts. Following the notation of Dechter *et al.* [1991], a *labeling*, ℓ, of a DTP, $\mathcal{D}$, is an STP formed by selecting one meta-value (disjunct) for each meta-variable (disjunctive constraint). A schedule s, then, is a solution to $\mathcal{D}$ if and only if it is a solution to one of $\mathcal{D}$'s labelings. For general DTPs, there are $O(k^{|C_{DTP}|})$ possible labelings, each of which must be explored in the worst case, making the DTP an NP-hard problem [Stergiou and Koubarakis, 2000]. In this particular example problem, of the $2^3 = 8$ possible labelings, only two ($A \rightarrow B \rightarrow C$ and $B \rightarrow A \rightarrow C$; which correspond to the STPs in Figure 1 (a) and (b) respectively) satisfy all scheduling constraints.

A *singleton* constraint, $c \in C^{k=1}_{DTP}$, is one that contains only a single disjunct. Tsamardinos and Pollack [2003] note that the subset $C^{k=1}_{DTP} \subseteq C_{DTP}$ can be used to form an STP $\langle V, C^{k=1}_{DTP} \rangle$. This STP can be used to compile new and tighter singleton constraints that constrain which meta-values can be assigned to which meta-variables. This forward-checking procedure prunes any disjunct that is inconsistent with the STP compilation, since it is guaranteed to be inconsistent with

the overall DTP. This pruning process may result in more constraints being added to the set $C_{DTP}^{k=1}$, which further tightens the STP compilation, possibly leading to more pruning. In the extreme, this process could prune until all disjunctive temporal constraints are singleton, eliminating the need for combinatorial search in the meta-CSP.

More generally, the meta-CSP formulation leads to a search algorithm that interleaves the STP forward-checking procedure with an assignment of a meta-value to a meta-variable. This has the effect of growing the set $C_{DTP}^{k=1}$ and incrementally tightening the corresponding STP compilation. If a particular assignment of a meta-value d_i to a meta-variable c_i leads to an inconsistent STP instance, that assignment is backtracked. Since at this point d_i is known to be inconsistent with the current STP compilation, a procedure known as *semantic branching* allows the STP relaxation to be tightened by adding d_i's inverse implication. Expressing these otherwise implicit constraints further tightens the STP relaxation, which in turn can lead to improved forward checking performance. Additionally, Tsamardinos and Pollack [2003] describe how to incorporate CSP techniques such as no-good recording and backjumping into the meta-CSP search algorithm to further decrease DTP solution algorithm runtime.

2.3 Temporal Constraint Satisfaction Problem

A Temporal Constraint Satisfaction Problem(TCSP), $\mathcal{T} = \langle V, C_{TCSP} \rangle$, is a special case of a DTP ($C_{STP} \subseteq C_{TCSP} \subseteq C_{DTP}$ and thus $STP \subseteq TCSP \subseteq DTP$) where each constraint $c \in C_{TCSP}$ takes the form $v_j - v_i \in [-b_{ji}^1, b_{ij}^1] \vee \ldots \vee [-b_{ji}^k, b_{ij}^k]$. That is, all disjuncts specify bounds over the difference between the *same* two timepoints. Similarly to the DTP, each of $O(k^{|C_{TCSP}|})$ possible labelings may need to be explored before finding a solution, making the TCSP an NP-hard problem [Dechter *et al.*, 1991]. While there exist procedures for transforming a DTP to a TCSP [Planken, 2007], the TCSP is more limited in the problems it can *naturally* represent. For example, the disjunctive constraints from the simple running example problem involve *different* pairs of variables (Table 1, lower), which the TCSP is not directly able to represent.

2.4 Minimality and Decomposability

Dechter *et al.* [1991] define both minimality and decomposability in terms of temporal constraint networks such as the STP and the TCSP. These definitions extend quite naturally to the DTP. A ***minimal*** constraint c_{ij} is one whose interval(s) correctly specify the set of *all* feasible values for the difference $v_j - v_i$. Similarly, a variable with a minimal domain is one whose constraints with the zero timepoint are minimal. A DTP is minimal if and only if all of its constraints and timepoint domains are minimal. A DTP is ***decomposable*** if any locally consistent assignment of a set of timepoint variables can be extended to a solution. Each of the STPs presented in Figure 1 are both minimal and decomposable.

In short, a minimal and decomposable DTP faithfully represents the complete and sound set of solutions by establishing the tightest bounds on timepoint variables such that: (1) no solutions are eliminated and (2) any assignment of a specific time to a timepoint variable (or bound to a constraint) that respects these bounds can be extended to a solution using a backtrack-free search. Establishing minimality and decomposability allows efficient processing of queries such as "at what time can activity X be performed" and "what are the potential relationships between activity X and Y". Unfortunately, establishing minimality and decomposability for general, disjunctive scheduling problems, such as the TCSP and DTP, is NP-hard [Dechter *et al.*, 1991].

The STP presents a special case where minimality and decomposability can be established efficiently (in $O(|V|^3)$) by applying an all-pairs-shortest-path algorithm, such as Floyd-Warshall [1962], to the distance graph to find the tightest possible path between every pair of timepoints, v_i and v_j, forming a fully-connected graph, where $\forall i, j, k, w_{ij} \leq w_{ik} + w_{kj}$. The resulting graph is then checked for consistency by validating that there are no negative cycles, that is, $\forall i \neq j$, ensuring $w_{ij} + w_{ji} \geq 0$ [Dechter *et al.*, 1991]. Recent work exploits sparsity in the constraint network to establish minimality and decomposability more efficiently [Xu and Choueiry, 2003; Shah and Williams, 2007; Planken *et al.*, 2008].

Since establishing minimality and decomposability in disjunctive temporal problems is NP-hard, much work has focused on efficient, polynomial-time methods to increase the level of consistency [Dechter *et al.*, 1991; Stergiou and Koubarakis, 2000; Tsamardinos and Pollack, 2003; Choueiry and Xu, 2004]. These consistency improvements are a partial step towards establishing minimal and decomposable representations and can be used as a preprocessing or constraint propagation technique during a meta-CSP search to find a component STP solution. Next we will prove that minimal, decomposable representations for consistent DTPs always exist.

3 The Existence of Minimal, Decomposable DTP Representations

In this section, we prove that the definitions of minimality and decomposability do indeed extend to DTPs by proving that a minimal and decomposable representation for a consistent DTP always exists. Despite the similarities between the TCSP and the DTP, there are challenges to extending the concepts of minimality and decomposability to the DTP. The heart of these challenges stem from the fact that DTP constraints can be specified over arbitrarily many different pairs of timepoint variables. While in a TCSP, it is well-defined whether or not there is a constraint that must necessarily be enforced between a pair of timepoint variables, this is not the case in DTPs. From the running example problem, the temporal difference constraint between A_{IN} and B_{OUT} only needs to be enforced if the temporal difference constraint between B_{IN} and A_{OUT} is not enforced, and *vice-versa*. That is, the structure of a minimal, decomposable temporal constraint network for a TCSP is obvious *a priori* (since disjunctive choices are always still between the *same* pair of timepoints), while the structure of a minimal, decomposable temporal constraint network for the more general DTP is not.

Notice that the set of solutions to both the DTP and TCSP can be represented as a set of minimal, decomposable STPs. To generate this set, we can naïvely enumerate each of the DTP's (exponentially many) feasible labelings, ℓ and then

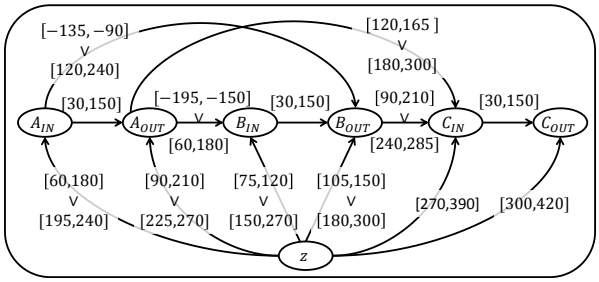


Figure 2: A TCSP representing the minimal network associated with the problem in Table 1.

calculate the minimal, decomposable STP associated with each ℓ. We exploit this observation in our proof that minimal representations of DTPs always exist, which follows as a series of corollaries to Dechter *et al.*'s proof (omitted for brevity) of Theorem 1:

Theorem 1. *[Dechter* et al.*, 1991] The minimal network, $\mathcal{M}$, of a given TCSP, $\mathcal{T}$, satisfies $\mathcal{M} = \cup_\ell M_\ell$, where M_ℓ is the minimal network of the STP defined by labeling ℓ, and the union is over all possible labelings.*

Corollary 1. *The minimal network, $\mathcal{M}$, of a given DTP, $\mathcal{D}$, satisfies $\mathcal{M} = \cup_\ell M_\ell$, where M_ℓ is the minimal network of the STP defined by labeling ℓ, and the union is over all possible labelings.*

Proof. Follows *mutatis mutandis* from Theorem 1. □

Corollary 2. *A minimal representation of a consistent DTP always exists.*

Proof. The minimal network, $\mathcal{M}$, of a DTP, $\mathcal{D} = \langle V, C \rangle$, can always be formulated as the TCSP, $\mathcal{T} = \langle V, C_\mathcal{M} \rangle$, where the set of constraints, $C_\mathcal{M}$, is composed of constraints $c_{ij} \in C_\mathcal{M}$ defined as $v_j - v_i \in \cup_\ell (\mathcal{M}_\ell)_{ij}$, where $(\mathcal{M}_\ell)_{ij}$ corresponds to the bound interval on the difference between v_j and v_i in the minimal network of the STP corresponding to label ℓ. □

Figure 2 represents the TCSP that forms the minimal network for the example problem presented in Table 1. Notice that we include every edge that shows up in either of the solution STPs. For example, the edge between A_{IN} and B_{OUT} is included with both a negative and positive interval, capturing the cases where A occurs before B and *vice-versa*, respectively. Note that in general an algorithm that assigns timepoints using the minimal network alone does is *not* guaranteed decomposability. For example, if we assign the duration of any of the activities represented in Figure 2 to 80, our future assignments should be limited according to selecting the STP in Figure 1 (a); however, the propagation of constraints in the minimal network alone does not guarantee that this would occur.

While the original definition of the decomposability property — a temporal network where any locally consistent assignment of a set of timepoint variables can be extended to a solution — extends naturally to the DTP, it is immediately not obvious whether or not decomposability can always be established for a DTP for a couple of reasons. First is that a DTP involves constraints with high cardinality, which, in general, can be much harder to decompose than a problem exclusively containing binary constraints [Gent *et al.*, 2000]. A second, related reason is that how the set of timepoint variables is assigned influences whether or not certain temporal difference constraints need be enforced in the set's complement. However, we again exploit our naïve representation to prove:

Theorem 2. *A decomposable representation of a consistent DTP always exists.*

Proof. If a DTP is consistent, its set of solutions can be represented as a set of minimal, decomposable STPs corresponding to each feasible labeling, ℓ. Given this representation, any assignment to a set of variables locally consistent with at least one of these STPs, by definition of a decomposable STP, is guaranteed to be extensible to a solution. □

This theorem leads naturally to a procedure for assigning variables in a locally consistent way. First, a variable can only be assigned a value if it appears in its domain (or alternatively, a constraint can only be assigned a bound within one of its intervals of possible bounds) in at least one minimal, decomposable solution STP. Second, once an assignment is made, all STPs that are inconsistent with this assignment are pruned, thereby guaranteeing that subsequent assignments will be locally consistent within one or more minimal, decomposable solution STPs.

In this section, we demonstrated that minimal, decomposable representations always exist for consistent DTPs. However, in general, enumerating every minimal, decomposable STP for each of the exponentially-many consistent labelings is neither compact nor efficient to reason over. In the next section we explore the challenges for more efficiently representing and establishing minimal, decomposable DTPs.

4 An Efficient Minimal, Decomposable DTP Representation

While we have shown that minimal, decomposable representations for consistent DTPs always exist, all representations are not necessarily created equally. As discussed earlier, one of the main advantages of a minimal, decomposable DTP representation is to support *queries* that ask "when can I perform activity X" either in general, or relative to another timepoint. However, presumably if one is interested in posing such queries, one is also interested in both making informed scheduling decisions, and perhaps performing additional queries in the future. So in this sense, we are also interested in how efficiently a minimal, decomposable DTP can be *updated*. Finally, given the exponential nature of the problem, in many scenarios, *space* requirements of the representation may also be of concern. In summary, we can compare representations in terms of (1) *query efficiency*, (2) *update efficiency*, and (3) *space efficiency*.

While our construction of the minimal network related to a DTP may support efficient queries, overall, using a possibly exponential number of STPs to represent and maintain (e.g. update) a decomposable DTP is likely to be quite inefficient in general. These issues lead to a natural question: are there

better ways for establishing and representing minimal, decomposable DTPs than the naïve approaches described in Section 3? We turn to some important related work in dispatching disjunctive schedules for insights into answering this. While not all goals of dispatchable execution align perfectly to those of maintaining minimal, decomposable DTPs, a dispatch agent must make fast recommendations (query efficiency, update efficiency) and may also have limited space capabilities (space efficiency).

4.1 Related Work: Fast Dispatch of Disjunctive Schedules

A minimal, decomposable STP instance naturally lends itself to *dispatchable execution* — an online approach whereby a dispatcher efficiently adapts to scheduling upheavals by imposing additional, restrictive constraints by scheduling timepoints immediately prior to execution [Muscettola *et al.*, 1998]. Shah and Williams [2008] generalize this idea to disjunctive scheduling problems by calculating a *dispatchable* representation of a TCSP. While a previous approach for dispatching DTPs exists [Tsamardinos *et al.*, 2001], this approach is similar in spirit to the naïve approach described in Section 3 by calculating and maintaining the set of all solution STPs. However, Shah and Williams [2008] recognize that many of the solution schedules contain significant redundancy, and so compactly represent the set of decomposable STP solution instances in terms of just their differences. This approach leads to not only a much more compact representation, but also faster execution by avoiding the need to simultaneously and separately update each disparate STP. The algorithm propagates each disjunct using a recursive, incremental constraint compilation technique, and for each disjunct, a list of logical conclusions is kept. Additionally, a list of conflicts is maintained as inconsistencies arise. Their basic execution dispatch algorithm adds each timepoint without any predecessors to an event list, and then as one of these timepoints becomes 'live' (when the current time falls within timepoint's domain), it is selected and updated to occur at the current time, after which the update is propagated throughout the remaining problem. They demonstrate empirically that their approach leads not only to orders of magnitude more compact representations, but also to orders of magnitude faster execution than the suggested naïve approach.

4.2 Extending to the DTP

Since the set of solutions for both DTPs and TCSPs can be represented by a set of minimal, decomposable STP solutions, Shah and Williams' approach for compactly representing the set of solutions by eliminating redundant information is naturally extensible to the DTP. In fact, note that the compact list of implied relationships (of the form $v_j - v_i \in [-b_{ji}, b_{ij}] \rightarrow v_y - v_x \in [-b_{yx}, b_{xy}]$) and conflicts (of the form $\neg v_j - v_i \in [-b_{ji}, b_{ij}] \vee \ldots \vee \neg v_y - v_x \in [-b_{yx}, b_{xy}]$) output by Shah and Williams' approach requires the generality of a DTP constraint to represent (since they are constraints involving *different* pairs of variables). This leads to a more general observation:

Observation 1. *Efficiently representing a set of minimal, decomposable STPs in conjunctive normal form (CNF) requires the representational power of a DTP.*

Constraints for Figure 1 (a)	**Constraints for Figure 1 (b)**
$A_{OUT} - B_{IN} \leq -60 \rightarrow$	$B_{OUT} - A_{IN} \leq -90 \rightarrow$
$(B_{OUT} - A_{IN} \leq -90 \vee)$	$(A_{OUT} - B_{IN} \leq -60 \vee)$
$A_{IN} - z \leq 180;$	$z - A_{IN} \leq -195;$
$A_{OUT} - z \leq 210$	$z - A_{OUT} \leq -225;$
$z - B_{IN} \leq -150$	$B_{IN} - z \leq 120;$
$z - B_{OUT} \leq -180$	$B_{OUT} - z \leq 150;$
$A_{OUT} - C_{IN} \leq 165;$	$C_{IN} - A_{OUT} \leq -180;$
$C_{IN} - B_{OUT} \leq -240;$	$B_{OUT} - C_{IN} \leq 210;$
	$A_{OUT} - A_{IN} \leq 75;$
	$B_{OUT} - B_{IN} \leq 75;$
	$C_{OUT} - C_{IN} \leq 75;$
	$z - C_{IN} \leq -345;$
	$z - C_{OUT} \leq -375;$

Table 2: A minimal, decomposable representation of the example logistics problem.

In addition to the constraints displayed in Figure 2, the constraints presented in Table 2 guarantee both minimality and decomposability for the example logistics problem. Following the same principle as Shah and Williams, we capture these implied relationships compactly by noting that the only difference in the labelings between the Figure 1 (a) and Figure 1 (b) are the labels $A_{OUT} - B_{IN} \leq -60$ (which implies the STP encoded by the temporal difference constraints in the first column of Table 2) and $B_{OUT} - A_{IN} \leq -90$ (which implies the STP encoded by the temporal difference constraints in the second column of Table 2). As a result, any update that is not common to both solution STPs will result in the the correct labeling being applied during forward-checking, which in turn results in a decomposable, minimal network after propagating the constraints. Whereas our representation required a linear (in the number of STP solution edges) number of additional constraints, to generally represent the two solutions encoded in Figure 1 in CNF form would require representing a combinatorial number of additional disjunctive constraints.

4.3 Open Challenges

To this point, we have exploited the DTP's similarity to the TCSP to address many important challenges associated with maintaining minimal, decomposable DTP representations. Particularly, we have shown that we can conceptually extend Shah and Williams' framework to calculate minimal, decomposable DTP representations, but minimality and decomposability only guarantee query efficiency. We now identify important differences between the DTP and TCSP that lead to unique challenges in efficiently establishing and maintaining minimality and decomposability in DTPs.

At a high level, Shah and Williams' approach is roughly similar to the meta-CSP search described in Section 2.2, however there are many important key differences that pose unique challenges for update efficiency. First, when propagating constraints in a TCSP, forward-checking only directly affects the constraint that is currently being propagated. However, in a DTP, pruning a meta-value during forward-checking could lead to a unary meta-variable (disjunctive constraint with only one remaining feasible temporal difference constraint) [Stergiou and Koubarakis, 2000]. This in turn could lead to a new constraint being posted in a distant, non-neighboring portion of the temporal network. That is, propagation jumps around the temporal network rather than only following links through

the network. A second, related difference is that DTP search decisions themselves also can fundamentally change the structure of the temporal constraint network by deciding to add temporal difference constraint between one pair of timepoints instead of some other, different pair. Both of these differences pose challenges to update efficiency, since it is no longer possible to systematically propagate constraints using the temporal network alone. Instead, updates must be simultaneously propagated through both the low-level, minimal temporal network and also the high-level, meta-level CSP. Incorporating support for general, decomposability in the meta-CSP requires fundamental changes to Shah and Williams' algorithm to efficiently learn and maintain all implied relationships and no-good constraints at the meta-level.

Together, these two differences pose a third challenge: can we encode the differences between two solution STPs as compactly, when the differences between them are more systematic in nature? Recall that Shah and Williams' basic approach attempts to exploit redundant information. However, when search decisions and constraint propagation lead to temporal networks with significant differences in structure (e.g., STPs that contain different constraints between different pairs of timepoints), it is unclear how compact, in general, the representation that this approach generates will be.

Finally, up to this point, when we discuss update efficiency, we have largely meant returning a perturbed minimal, decomposable DTP back to a minimal and decomposable state. However, in many applications, the time it takes to establish the initial minimal, decomposable representation may be critical. Shah and Williams' approach is intended as a precompilation approach, and so is never evaluated in terms of initial compilation time. We conjecture that incorporating recent CSP-based search techniques, such as forward-checking, no-good learning, etc. [Tsamardinos and Pollack, 2003], and also SAT-based techniques, such as unit-propagation and a two-literal watching, etc. [Armando *et al.*, 2004; Nelson and Kumar, 2008] could speed the overall process of establishing a decomposable by orders of magnitude, but also poses significant algorithmic engineering challenges.

5 Discussion

In this paper, we demonstrated how the concepts of minimality and decomposability, which had previously been formally defined for only the STP and TCSP, can naturally be extended to the more general DTP formulation to represent the complete, sound set of solutions. We contributed proofs that minimal and decomposable representations of consistent DTP always exist, though not necessarily in an efficient, compact form. We introduced metrics for comparing different minimal, decomposable DTP representations in terms of efficiency, including compactness (space) and query and update speed. We then both discussed the challenges of and offered insights for extending an approach for incrementally compiling TCSPs for fast schedule dispatching to the more general DTP. An interesting extension of this work would be an algorithm that can compute and maintain minimal and decomposable DTP representations in a more incremental or anytime manner for applications that cannot afford to wait for costly precompilation algorithms to complete. Our vision is that the insights of this work can lead to online algorithms that effectively and efficiently adapt to scheduling eventualities that arise in real time and do so with minimal space requirements. These algorithms would have significant implications for logistics applications where the pace of scheduling perturbations may outstrip a scheduler's ability to replan or reschedule, while also granting more autonomy for practitioners to, on-the-fly, select the schedules and plans that best suit their immediate needs and preferences.

References

[Armando *et al.*, 2004] A. Armando, C. Castellini, E. Giunchiglia, and M. Maratea. A SAT-based Decision Procedure for the Boolean Combination of Difference Constraints. In *Proc. of SAT'04*, pages 166–173, 2004.

[Boerkoel and Durfee, 2011] J.C. Boerkoel and E.H. Durfee. Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem. In *Proc. of AAMAS 2011*, pages 141–148, 2011.

[Choueiry and Xu, 2004] B.Y. Choueiry and L. Xu. An efficient consistency algorithm for the temporal constraint satisfaction problem. *AI Communications*, 17(4):213–221, 2004.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. In *Knowledge representation*, volume 49, pages 61–95. The MIT Press, 1991.

[Floyd, 1962] R.W. Floyd. Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[Gent *et al.*, 2000] I. Gent, K. Stergiou, and T. Walsh. Decomposable Constraints. *Artificial Intelligence*, 123(1-2):133–156, 2000.

[Muscettola *et al.*, 1998] N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating temporal plans for efficient execution. In *Proc. of KR'98*, pages 444–452, 1998.

[Nelson and Kumar, 2008] Blaine Nelson and T. K. Satish Kumar. CircuitTSAT: A solver for large instances of the disjunctive temporal problem. In *Proc. of ICAPS-08*, pages 232–239, 2008.

[Planken *et al.*, 2008] L. Planken, M. de Weerdt, and R. van der Krogt. P3C: A new algorithm for the simple temporal problem. In *Proc. of ICAPS-08*, pages 256–263, 2008.

[Planken, 2007] Leon R. Planken. Temporal reasoning problems and algorithms for solving them (literature survey). Literature survey, Delft University of Technology, October 2007.

[Shah and Williams, 2007] J.A. Shah and B.C. Williams. A Fast Incremental Algorithm for Maintaining Dispatchability of Partially Controllable Plans. In *Proc. of ICAPS-07*, 2007.

[Shah and Williams, 2008] J.A. Shah and B.C. Williams. Fast Dynamic Scheduling of Disjunctive Temporal Constraint Networks through Incremental Compilation. In *Proc. of ICAPS-08*, 2008.

[Stergiou and Koubarakis, 2000] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

[Tsamardinos and Pollack, 2003] I. Tsamardinos and M.E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151(1-2):43–89, 2003.

[Tsamardinos *et al.*, 2001] I. Tsamardinos, M.E. Pollack, and Ganchev P. Flexible Dispatch of Disjunctive Plans. In *Proc. of ECP-06*, pages 417–422, 2001.

[Xu and Choueiry, 2003] L. Xu and B Choueiry. A new effcient algorithm for solving the simple temporal problem. In *Proc. of TIME-ICTL-03*, pages 210–220, 2003.

Timeline-based Planning System for Manufacturing Applications

Minh Do and **Serdar Uckun**
Embedded Reasoning Area, Palo Alto Research Center.
Email: {minh.do,uckun}@parc.com

Abstract

In recent years, the Embedded Reasoning Area (ERA) has been developing a planning system targeting fast online planning problems in manufacturing. The planner, which is based on general-purpose AI planning techniques, has evolved through several iterations and successfully solved applications such as hyper-modular printers, modular packaging machines, material control for LCD manufacturing, and warehouse management. In this paper, we will describe the core techniques underlying the current version of the planner: a combination of timeline-based state representation and action-based planning algorithm. This combination is proven to be flexible and can quickly adapt to new applications and at the same time can scale to complex problems.

1 Introduction

Our research on model-based online planning starts with the Tightly Integrated Parallel Printer (TIPP) project [Ruml *et al.*, 2005; Do *et al.*, 2008; Ruml *et al.*, 2011] where we need to effectively control reconfigurable printing systems. After the success of this project, there have been efforts in adopting the software, in particular the planner to new applications. The first application was controlling modular packaging machine, which shown that the adaptation of the TIPP planner can effectively control (in simulation) a variety of high-speed infeed systems of food flow-wrapper machines. However, this is just the first step in generalizing it to solve a more general class of problems in manufacturing.

After the initial investigation in the packaging domain, we have been further extending our model-based planner so that it can easily be adapted to a wide variety of application domains. Recently, our planner has been used in several funded projects by the IHI Corporation in 2010 and 2011 for different applications: Material Control System (MCS) and Automated Warehouse. In this paper, we outline the architecture of the new planner and the application domains that it was tested on.

The rest of this paper is organized as follows: we start with the timeline-based online planning architecture in the next section. We then follow with two implemented planning algorithms (1) forward state-space; and (2) partial-order in Section 2.3 and Section 2.4. We outline the results of using our planner in the manufacturing applications outlined above and we finish the paper with some future work.

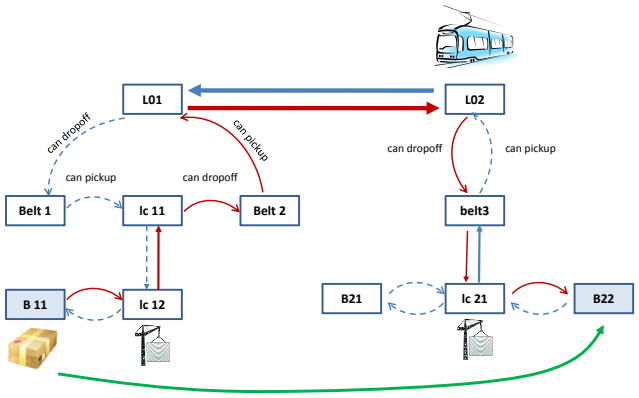


Figure 1: A logistics example

2 Overall Architecture

Planning is the problem of finding a (sequential or parallel) sequence of actions that when executed from a known initial state will achieve all pre-defined goals. Our group has been working on "fast continual on-line planning" problems where user's goals and system updates continuously arrive in concurrent with plan executions of previously found plans. In *fast* we mean the software generally needs to find a complete solution within a few seconds (sub-second in several cases).

Our current Plantrol planner uses a timeline-based planning approach that operates by continually maintaining the *timelines* that capture how different system state variables change their values over time. The planner builds and maintains consistent plans by adding *tokens* to the affected timelines; with each token represents a different operation/change affecting the state variable represented by that timeline. The overall framework allows selection among multiple planning algorithms, all share the same timeline-based state representation, for a given task. In turn, different planning algorithms can call different search algorithms and constraint solvers (e.g., temporal reasoning, uncertainty reasoning) to solve either planning or replanning tasks effectively.

To illustrate different concepts, we will first present a simple example that will be used throughout the paper:

Example: shown in Figure 1 is an example inspired by IHI's MCS application. In this example, a package located at location *B11* needs to be moved to *B22* using first the crane located at *LC12*, then the overhead vehicle (OHV) that is originally at *L02* and then lastly the second crane originally located at *LC21*. The arrows in solid red color show the path of the package. Note that there are a couple of actions belong to a final plan but are not included in this path such as moving the OHV from *L02* to *L01* and the second crane from *LC21* to

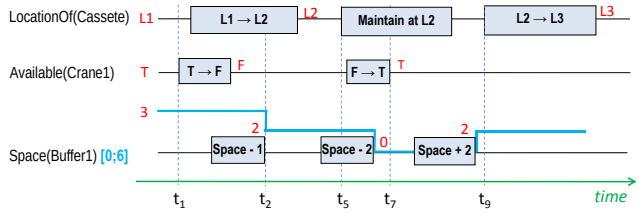


Figure 2: Timeline example

Belt3. They are represented by solid blue arrows. The remaining dotted blue arrows represent the other actions available but are not part of the final plan.

2.1 Timeline-based State Representation

The input to the deterministic online planner consists of:

1. a set of variables V with each $v \in V$ is associated with a given domain of values $D(v)$;
2. a set of actions A, each specified by its *(pre)condition* and *effect* lists. An action condition represents a constraint on the value of a given variable (e.g., $v = x$) and an action effect represents a change to the value of a given variable (e.g., $v \leftarrow y$).
3. a complete variable assignment for all $v \in V$ represents the fully observable initial state I.
4. a partial variable assignment G represents the desired goal condition.

An action a is applicable in state s if all of its conditions are satisfied by s and the resulting state from applying a in s reflects the changes caused by a's effects on s. The planner needs to find a consistent sequence of actions (a plan) P that can connect I to G.

For *online continual* planning scenarios, finding and executing plans and goal-arriving are interleaved. Given that the planner needs to continuously reason about those interleaving processes, we need to effectively maintain the status of different state variables as they change values over time. One good way to do so is through timelines and there are several application-oriented planners, including TIPP, that have used this approach at different levels [J. Frank, 2000; Fratini *et al.*, 2008].

Figure 2 shows an example of the timelines of several variables in our leading example: (1) a *multi-value* (discrete) variable $v_1 = LocationOf(Package)$ that represents the package location; (2) a *binary* variable $v_2 = Available(Crane1)$ represents whether or not *Crane1* is busy carrying some package; (3) a *continuous* variable $v_3 = Space(Buffer1)$ represents the available/empty space in *Buffer1*. While we currently only support three types of variables (which are most common) in our planner, theoretically any variable with a certain value domain can be included in the timeline set managed by the planner.

The timeline for a given variable v consists of a value $c_v \in D(v)$, which is the value of v at the current wall-clock time t_c and a set of *tokens* representing future events affecting the value of v. Those events represent pre-committed assignments of different equipments/resources/objects. Figure 2 shows one example where there are three tokens in the timeline for $v_1 = LocationOf(Package)$ representing the following events (in this order): (1) the value of v_1 changes from the current value $v_1 = L_1$ to a new location $v_1 = L_2$, (2) and $v_1 = L_2$ needs to be maintained for certain duration; then (3) it changes again from L_2 to L_3. Each token tk is represented by:

- Start and end time points $start(tk)$ and $end(tk)$.
- A start value v_s (or bounds on start value $[lb, ub]$ with $lb \leq ub$ for continuous variable).
- Start condition (e.g., $v = v_s$) specifies the condition that needs to be satisfied by the token. Right now, we support: $=, \neq, >, <, \geq, \leq, NONE$.
- Change operation $\langle operator, value \rangle$ (e.g., $v \leftarrow v + 5$ or $v \leftarrow x$) specifies how the variable value is changed within the token duration. Some change operators are: $\leftarrow$, +=, -=, $\times$=, /=, *CHANGE, USE, MAINTAIN*[1].

Given that tokens generally represent conditions and changes caused by actions, there can be temporal relations between tokens that are either: (1) conditions/effects of the same action a; (2) conditions/effects of actions that are related to each other. For example, before we *move* the package from $L1$ to $L2$ using $Crane1$, the crane needs to *pick_up* the package first. Thus, tokens caused by the *pick_up* action need to finish before the tokens added by the *move* action and thus there are temporal orderings between them.

Figure 3 shows an example of tokens on different timelines created by a given action instance. On the left side, we show the action representation in PTDL, a variation of PDDL [Fox and Long, 2003] – a standard planning modeling language, and the right side shows five tokens which would be added to different timelines if action $move$ is added to the plan. The same action starting time point t_s will be the starting time of four tokens and thus those four are constrained to start together.

For a given action a, we will use $T(a)$ to denote the set of tokens caused by a.The set of timelines for all variables is *consistent* if:

- *Value consistent:* Consecutive tokens on the same timeline should make up a consistent sequence of changes. Thus, the end value of a given token should *match* with the start value of the next token[2].
- *Temporal consistent:* All temporal constraints between tokens should not cause any temporal inconsistency. One example of temporal inconsistency is that two temporal orderings: $t_1 < t_2$ and $t_2 < t_1$ are both deductible from the temporal network.

A consistent timeline for v_g achieves a given goal $g = \langle v_g, x \rangle$ (i.e., $v_g = x$) at the end of the timeline for v_g if the end value of the last token matches with x. Alternatively, we say that it achieves g at some point in time if there exist a token T such that the end value of T matches x. For a given goal set G, if for all $g \in G$ the consistent timeline for v_g satisfies g then we say that the set TL of all timelines for all variables satisfy G or $TL \models G$.

2.2 Timeline-based Online Continual Planning

The previous section discusses how the world state is represented and maintained in continual planning by using a set of evolving timelines containing tokens representing actions'

[1]The variable value at the end of the token is calculated based on the start value and the change operation.

[2]In *matching*, we generally mean equal but for continuous variables that are represented by a $[lb, ub]$ interval, matching means that two intervals overlap.

```
(:action move
 :parameters (?v - vehicle ?l1 ?l2 - location)
 :duration (/ (distance ?l1 ?l2) (speed ?v))
 :condition
       ([start,end] (direct-connect ?l1 ?l2))
 :effect
       (over-all
             (change (location-of ?v) ?l1 ?l2)
             (use (path ?l1 ?l2)))
       ([start, start + 2] (change (space-free ?l1) F T))
       ([end - 2, end] (change (space-free ?l2) T F)))
```

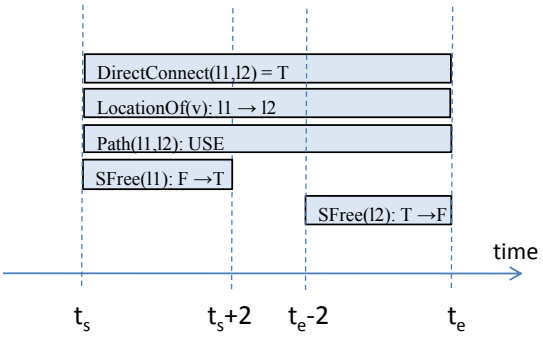


Figure 3: Action and its corresponding tokens

conditions and effects. In this section, we provide a high-level planning algorithm that operates on timelines and finds consistent plans.

Algorithm 1: Timeline-Based Planning Algorithm

input : A consistent timeline set TL, a goal set G
output: Plan P achieves G & an updated timeline set TL

1 Let: $P_0 \leftarrow \emptyset$, $TL_0 \leftarrow TL$, and $s_0 = \langle P_0, TL_0 \rangle$;
2 Initialize the state set: $SQ = \{s_0\}$;
3 **while** $SQ \neq \emptyset$ *and* **done** = false **do**
4 Pick the best state $s = \langle P_s, TL_s \rangle$ from SQ;
5 **if** TL_s *is consistent and* $TL_s \models G$ **then**
6 **done** = *true*
7 **else**
8 Generate zero or more revisions P' of P_s;
9 Generate timeline sets $TL' \leftarrow TL \bigcup T(P')$;
10 Add temporal constraints between temporally related tokens in TL';
11 Add $s' = \langle TL', P' \rangle$ to generated state set SQ;
12 Execute P_s;
13 Revise the master timeline set: $TL \leftarrow TL_s$;

Algorithm 1 shows at a high-level a planning algorithm operating on timelines. Some notations used in this algorithm, and all subsequent algorithms described in the next several sections are:

- For each time point tp (e.g., token's start/end time-point): $est(tp)$ and $lst(tp)$ represent the earliest and latest possible times that tp can happen.
- For an action set A: $T(A)$ is the set of tokens caused by all actions in A. Similarly, $T(P)$ is the set of tokens caused by all actions in the plan P.

The planner starts with a consistent timeline set TL representing all changes and constraints related to all state variables from the current wall-clock time. It needs to find a plan P such that (1) adding $T(P)$ to TL does not cause any inconsistency, (2) achieve all goals, and (3) executable (i.e., all tokens caused by this plan should be able to start after the wall-clock time at which the plan is found). The planner starts with an empty plan and keeps revising it until achieving these objectives (lines 8-11). The planner tries to find the *best* plan by maintaining a set of generated states (which is composed of a plan P and the timelines resulted from adding tokens caused by P to the original timelines) and at each step picks the best from the generated set to check for being a valid plan. When the best plan P is found, we execute P (line 12) and incorporate its effects in the continually maintained timelines (line 13).

This high-level algorithm obviously lacks many details such as: how to revise P_s (line 8)? what is the *best* plan? or what exactly is the representation of the plan during the planning process? On the other hand, it's general enough to capture both systematic and local-search style of planning, and for different planners that can handle different set of variables and constraints. In the next two sections, we describe two implemented algorithms based on this framework.

2.3 Forward State-Space Planner on Timeline

Forward state-space (FSS) planners move forward in time through fixed-time complete state. It starts with an empty plan and gradually add actions at some fixed wall-clock time to the end of the currently expanding partial plan until the final sequence of actions satisfies the goals. In short, a visited "planning state" s of a FSS planner consists of: (1) a time-stamp t_s of s; (2) a set of timelines in which all tokens (i) end after t_s and (ii) have fixed start and end times.

The algorithm starts searching from the current wall-clock time t_c but will execute the plan at the (expected) wall-clock time $t_e > t_c$ when the plan is found. To start the planning process, the planner "freezes" all tokens in all timelines and remove all tokens that end before t_e. This step simplifies the token and timeline representation and also reduces their sizes. The key details here are the successor generating functions to create subsequent search nodes:

- *Applicable*: for each action a, the FSS planner moves forward in time from the current state's time stamp t_s until it finds an earliest time $t_a \geq t_c$ that if a executes at t_a then all new tokens added will not cause any inconsistency. Any action t_a that we can find a *consistent* execution time t_a is added to our candidate set.
- *Apply*: the planner generates successors by creating tokens corresponding to action's conditions and effects and add them to the current timelines.
- *AdvanceTime*: this is a special action that helps move the state time-stamp t_s forward closer to the goal. When moving the time-stamp forward, it basically sets the newer lower-bound on the future action execution time and thus: (1) simplifying the timelines (remove all tokens finish before the new time-stamp); and (2) reducing the interactions between existing tokens and future actions.

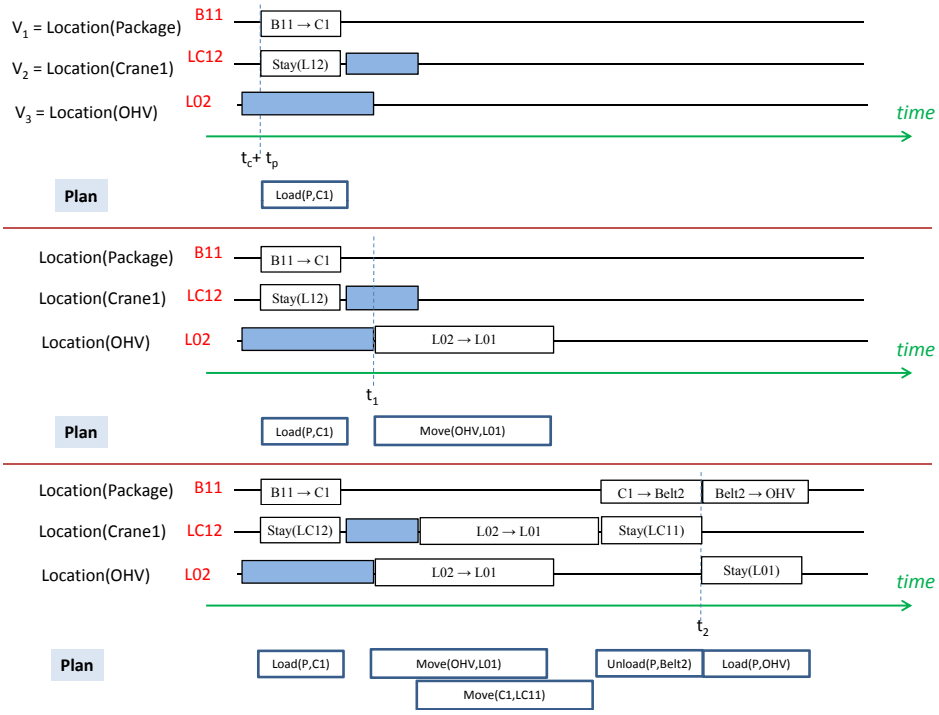


Figure 4: Example illustrating several steps of the FSS on timeline algorithm: adding actions in "forward" direction.

Given that the plan returned by the FSS algorithm has all actions and tokens tied to some fixed wall-clock times, depending on a particular search algorithm and heuristic setting, the FSS planning algorithm may not return the plan with all actions start at the earliest possible time according to their temporal relation. The "fixed-time" plan found by the FSS algorithm can easily be converted to flexible temporal plans using techniques described in [Do and Kambhampati, 2003].

Figure 4 shows several steps with the goal of having a package inside an OHV. In the timelines for the three variables mentioned above, tokens represented by solid rectangles are from previous planning episodes and thus tokens created by the current planning process should not overlap with them. We start by setting up the time stamp t_e, and the planner starts by adding an action of loading the package into *Crane 1* at t_e. This action addition creates two fixed-time tokens on the timelines for v_1 and v_2. We then apply the *AdvanceTime* action (several times) to reach t_1 and apply the second action to move the OHV to $L01$ (this adds one token to the timeline of v_3). After several steps of adding regular actions (e.g., $Move(Crane, LC11)$, $Unload(P, Belt2)$) and several *AdvanceTime* actions, we *load* the package into OHV. At this time, all timelines are consistent and achieving all goals so we terminate the planning process.

2.4 Partial-Order Planner (POP) on Timeline

The FSS algorithm described in the previous section finds plans by moving forward through a sequence of consistent timelines until a given timeline set satisfying all goals. On the other hand, the POP algorithm finds plans by starting with an inconsistent timeline set and systematically refines it until it becomes consistent. The planner searches backward from the goals. For that, it first creates special tokens representing the goals and the planner's objective is to create enough tokens through action addition so that those goal tokens are all eventually supported. Instead of finding *Applicable* actions as in the FSS algorithm, it finds *Relevant* actions, which can contribute new tokens that support some currently un-supported tokens. We have two-level branching: (1) over actions that are deemed *relevant*; and (2) over token ordering where the new tokens introduced by the newly added actions can be added in the respective timelines. Note that there is no fixed starting time for all actions and tokens but their start/end times are represented by floating time points.

Figure 5 shows several steps in the POP algorithm finding the plan with the same set of actions as the FSS algorithm shown in Figure 4. The planner starts by creating a special token $v_1 = In(OHV)$ at the end of the timeline for v_1. It then adds an action $Load(P, OHV)$ to the plan because that action can add a token to support $v_1 = In(OHV)$. Appropriate temporal orderings are also added between related time points (we show some of them in the figure). The algorithm keeps picking un-supported tokens and add actions to support them until the timelines are consistent and the final plan is found.

FSS vs. POP: Two algorithms have distinctive advantages. The *fixed-time* and the association of a time-stamp for each search state during the planning process lead to:

- Smaller state representation: (1) any token ends before the current state's time-stamp can be removed from consideration; (2) no order between different tokens need to be stored. They are implicitly implied by the fixed start/end time of all tokens.
- Lower branching factor: each applicable action generates exactly one successor.

Therefore, the FSS planner likely find some valid plan faster. On the other hand, the POP algorithm employs a more

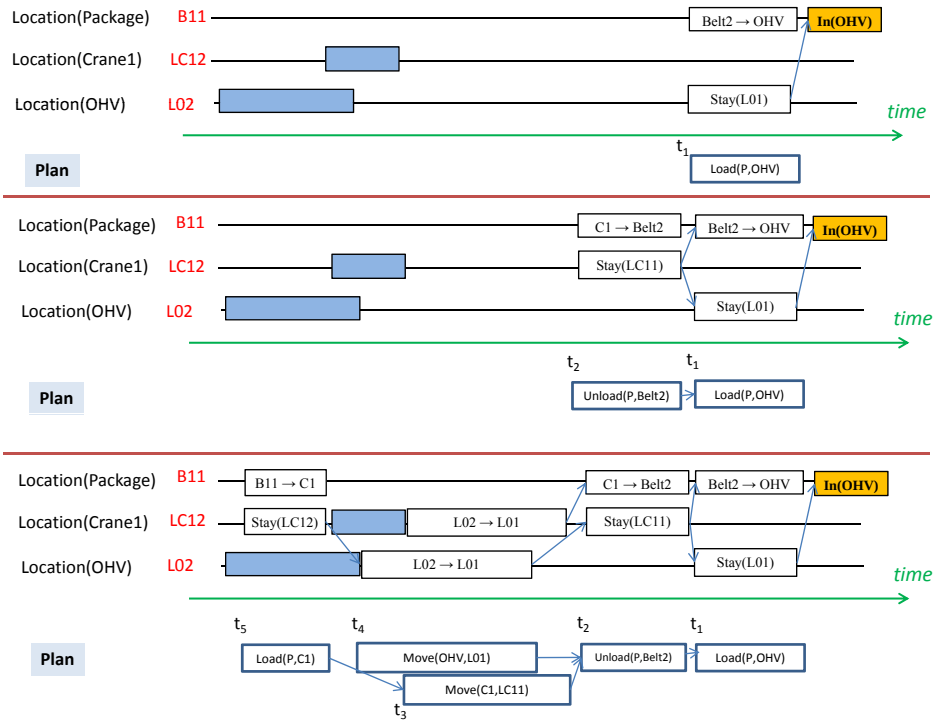


Figure 5: Example illustrating several steps of the POP on timeline algorithm: adding flexible actions in "backward" direction.

complete branching rule and thus does not rule out any valid solution. FSS planner, due to the fact that it doesn't consider all possible action starting times (by only moves the time-stamp forward to the next significant time point) may miss some solutions.

Depending on the actual application, one algorithm may be more appropriate: FSS is likely more suitable for applications where finding a plan quickly is of critical; and POP may be more appropriate where planning time is not critical but plan quality is more important.

2.5 Makespan-estimation Heuristic

Planner's performance, especially search-based, highly depends on the quality of the heuristic guiding its exploration of its solution space. In our targeted problems of Plantrol, we concentrate on finding plans that optimize for goal achievement time, which is highly related to *makespan* (i.e., plan execution time). Our heuristic is based on building the relaxed temporal planning graph (RTPG) and adjusting its estimation with the potential conflicts with tokens of the previous plan.

For each planing state s, the RTPG estimates the temporal distance between the current state in the search tree and the final state that the search algorithm tries to reach. In our FSS algorithm (Section 2.3), the heuristic estimates the distance between the current state and the goal state while in the POP algorithm (Section 2.4), the heuristic estimates the distance between the current state and the initial state. Given that the heuristic procedures for both types of planner are very similar, we will just discuss and give an example for the FSS planner.

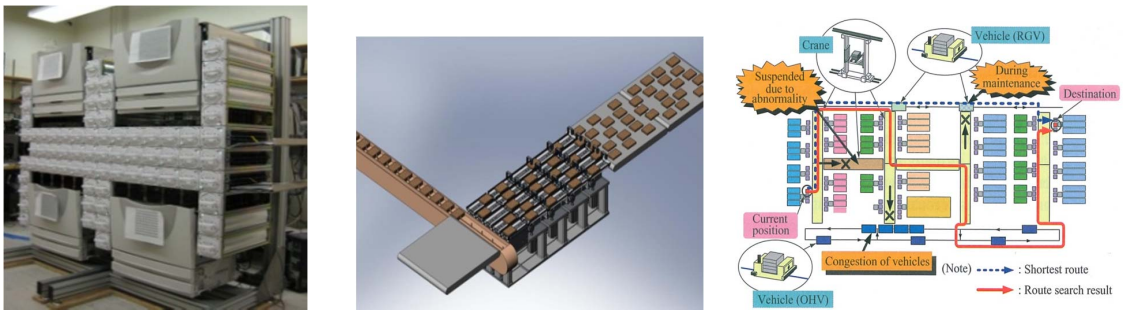
Given a timeline set TL representing a state during the planning process and the goal set G to be achieved, the algorithm will estimate the finishing time of a shortest plan that achieves G and built on top of TL (i.e. extends and includes all tokens in TL). The algorithm starts from the time-stamp $t = t_{TL}$ of TL and moves forward in a similar fashion to the FSS algorithm described in Section 2.3. However, instead of selecting which action to add next, we will optimistically apply all actions that have their conditions satisfied at t and ignore their conflicts. The neglect of conflicts between overlapping actions lead to the name *"relaxed" temporal planning graph*.

When actions with their conditions satisfied at time t are added, we add the tokens caused by their effects (refer to Section 2.1) to the collective pool of tokens that can lead to new values. At any given moment, we maintain the set D of (optimistically) achievable values, starting with the current values at t_{TL} in all timelines. After activating all actions having all of their conditions satisfied at time t and add tokens representing their effects into TL, we move forward (increase t) to the earliest end time t_e of any token in TL and add the new value achieved by all tokens ending in t_e to D. We repeat the process until either: (1) D contains all values of G; (2) there is no additional token in S to advance to its end time (and thus there is no new value to add to D).

3 Applications

Our online temporal planner has been successfully tested on several applications.

Tightly Integrated Parallel Printer (TIPP): The TIPP reconfigurable printer design allows building custom reconfigurable printer configurations from shared components. This project requires a software controller that can work with any design and it starts our work in the integrated planning and control framework. Our first planner built for this application uses the timeline representation for shared resources and combines it with the non-timeline state representation for logical variables. The planner works very well and can control two physical prototypes and hundreds of conceptual designs with



Parallel Printer **Modular Packaging** **Material Control System**

Figure 6: Example instances of the applications addressed by PARC's planner

the productivity of up to 220 page-per-minute (which requires planning/solving time to be less than 0.27 seconds). We have published extensively for this application [Ruml *et al.*, 2005; Do *et al.*, 2008; Ruml *et al.*, 2011].

Packaging: After the conclusion of the TIPP project, we investigated the application of our model-based planning+control technology to controlling an automated infeeder for a packaging line of food and consumer packaged goods. In this system, products arrive continuously at high-speed from the end of the production line and need to be arranged into a specific configuration for downstream primary and secondary packaging machines. In collaboration with a domain expert from the packaging industry, we developed an innovative design for a reconfigurable parallel infeed system using a matrix of interchangeable smart belts. We also adapted our online model-based Plantrol planner to this domain. Our planner can control various configurations of the new infeed system through simulation both in nominal planning and when runtime failures occur. We are also building a physical prototype to validate the new design and our software framework. More details are described in [Do *et al.*, 2011a].

Material Control System: Recently, in early 2010, we have successfully applied our planning framework to another application: planning for the Material Control System (MCS) of Liquid Crystal Display (LCD) manufacturing plant in a joint project between the Embedded Reasoning Area at PARC and the Products Development Center at the IHI Corporation. The model-based planner created at PARC was able to successfully solve a diverse set of test scenarios provided by IHI, including those that were deemed very difficult by the IHI experts. The short project time (2 months) proved that model-based planning is a flexible framework that can adapt quickly to novel applications. This the the first project where the the full timeline based representation, as described in this paper, was used for the planner. More details on the domain and the adaptation effort are described in [Do *et al.*, 2011b].

Automated Warehouse: Earlier this year, we collaborated with IHI again on another project on Automated Warehouse control[3]. The adaptation of our planner was able to successfully control a very large (few thousand objects) warehouse system with complex constraints. It consistently found plans up to hundreds of actions in less than one second. We hope to be able to describe the details in the future publication.

[3]Due to the proprietary IHI's warehouse design, we are not able to reveal the details or show any configuration example.

4 Conclusion & Future Work

In this paper, we introduce an automated planning framework for fast online continuous planning applications. The planner combines timeline-based state representation and action-based planning algorithms. The result planner has been used successfully in several manufacturing applications. We are currently working on extending both the expressiveness of our modeling language, adding supports for handling constraints such as uncertainties, and looking to apply our framework for even more applications.

References

[Do and Kambhampati, 2003] Minh Do and Subbarao Kambhampati. Improving the temporal flexibility of position constrained metric temporal plans. In *Proc. of ICAPS-03*, 2003.

[Do *et al.*, 2008] Minh Do, Wheeler Ruml, and Rong Zhou. On-line planning and scheduling: An application to controlling modular printers. In *Proc. of AAAI08*, 2008.

[Do *et al.*, 2011a] Minh Do, Lawrence Lee, Rong Zhou, and Lara Crawford. Online planning to control a packaging infeed system. In *Proc. of the Twenty-Third Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-11)*, 2011.

[Do *et al.*, 2011b] Minh Do, Kazumichi Okajima, Serdar Uckun, Fumio Hasegawa, Yukihiro Kawano, Koji Tanaka, Lara Crawford, Ying Zhang, and Aki Ohashi. Online planning for a material control system for liquid crystal display manufacturing. In *Proc. of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, 2011.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[Fratini *et al.*, 2008] S. Fratini, F. Pecora, and A Cesta. Unifying planning and scheduling as timelines in a component-based perspective. *Archives of Control Sciences*, 18(2):231–271, 2008.

[J. Frank, 2000] P. Morris J. Frank, A Jonsson. On reformulating planning as dynamic constraint satisfaction. In *Symposium on Abstraction, Reformulation and Approximation*, 2000.

[Ruml *et al.*, 2005] Wheeler Ruml, Minh B. Do, and Markus Fromherz. On-line planning and scheduling for high-speed manufacturing. In *Proc. of ICAPS-05*, pages 30–39, 2005.

[Ruml *et al.*, 2011] Wheeler Ruml, Minh Do, Rong Zhou, and Markus Fromherz. On-line planning and scheduling: An application to controlling modular printers. *Journal of Artificial Intelligence Research*, 40:415–468, 2011.

Supply Network Coordination by Vendor Managed Inventory – A Mechanism Design Approach

Péter Egri[1], **József Váncza**[1,2]
[1]Fraunhofer Project Center for Production Management and Informatics
Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13-17, 1111 Budapest, Hungary
[2]Department of Manufacturing Science and Technology, Budapest University of Technology and Economics
H-1111 Budapest, Egry József u. 1., Hungary

Abstract

The paper studies a generic coordination problem in supply networks with some retailers and a single supplier agent. The parties posses private information—the retailers on uncertain demand forecasts, the supplier on production costs—and seek to maximise their own utilities. So as to design a coordination mechanism that warrants the satisfaction of all market demand at maximal social welfare, we model the problem as a non-cooperative game. By relying on the conceptual apparatus of mechanism design theory, we provide an analytical explanation for the widely used vendor managed inventory (VMI) where the responsibility of planning and all the risks of over- and underproduction are at the supplier. After proving that under reasonable assumptions a fair sharing of these risks is not possible, we present a family of coordination mechanisms that are efficient and can be implemented without an individually existing mechanism. Beyond new managerial insights—like modelling VMI as a service where payment should also depend on the accuracy of information communicated—a novel approach is provided to handling supply networks consisting of autonomous agents.

1 Introduction

Supply networks are large and complex systems, characterised by the existence of numerous competitive agents, dynamic structures, uncertain knowledge and difficult planning and decision making problems. The uncoordinated actions in such a system lead to e.g., suboptimal performance, exemplified in a simple case by the well-known *prisoners' dilemma*. In supply networks the appearance of this phenomenon is called *double marginalisation*: since every enterprise concern their own profit when making decisions, the aggregate benefit is in general lower than if the enterprises were vertically integrated. This suboptimality also results in waste of materials, labour, energy, environmental resources and eventually causes significant financial losses for the enterprises.

In a vertically integrated supply network with multiple retailers and a supplier, centralising the replenishment and inventory management decisions at the supplier side is advantageous compared to the situation where each retailer has to decide individually. This centralisation approach is called *risk pooling*, and it is proved to result both in lower overall safety stocks and in lower average inventory levels [Simchi-Levi *et al.*, 2000].

In order to use the idea of risk pooling in vertically non-integrated networks, the *vendor managed inventory* (VMI) business model is applied frequently. In VMI the supplier takes all risks and full responsibility for managing a one-point inventory, while it has to fulfil the entire demand of the retailers, even if this requires additional costs due to extra capacity usage, overtime, outsourcing, or rush production orders [Simchi-Levi *et al.*, 2000]. This situation is clearly disadvantageous for the supplier, in fact, the main practical reason underlying VMI is the market power of the retailers, and not the mutual interest of the partners. Furthermore, since the retailers are not faced with the consequences of an imprecise forecast directly, they are not inspired to increase their efforts in accurate forecasting.

Sometimes the retailers have incentives even for distorting the forecasts. If the performance of the retailers are measured by the eventual shortage, then they tend to overplan demand and forward too optimistic plans towards the supplier. On the other hand, if the retailers are rewarded for overperforming the plans, then they tend to underestimate the demand. In both cases, the selfish distortion of information will introduce additional uncertainty into the demand forecasts, and lead to higher operational costs.

2 Related Work

Agents provide a natural metaphor for manufacturing in supply networks, where the knowledge is both incomplete (distributed) and imprecise (uncertain) [Egri and Váncza, 2007]. Unfortunately, most of the existing multi-agent models assume *benevolence*—i.e., the agents must implicitly share a common goal—, which holds in some situations, but certainly not in a supply chain of autonomous enterprises. Lack of benevolence can result not only in suboptimal behaviour, but also in the collapse of the production process, as an earlier study in decentralised production scheduling showed [Váncza and Márkus, 2000].

Therefore, the number of deployed multi-agent systems that are already running in real industrial environments is

unsurprisingly small. An other important reason for this is that in the behaviour of a multi-agent system there is always an element of *emergence* which can be a serious barrier to the practical acceptance of agent-based solutions. Industry needs safeguards against unpredictable behaviour and guarantees regarding reliability, safety and operational performance [Monostori *et al.*, 2006].

Such guarantees can be given by applying the results of game theory and *mechanism design* [Rosenschein and Zlotkin, 1994; Shoham and Leyton-Brown, 2008; van der Krogt *et al.*, 2008]. For example, the auction theory has already demonstrated its value with several applications, even in the electronic markets. This success is mainly due to the fact that these markets are *well-structured* with clear, exact regulations. Where the conditions of control are given by formal rules, introducing the concepts and apparatus of game theory is a really promising approach. That is why it has recently become popular for analysing the behaviour of automated agents operating on the Internet [Dash *et al.*, 2003; Nisan *et al.*, 2007].

Considering enterprises with own objectives also resulted in various game theoretic formalisations, both cooperative and non-cooperative ones. The former approach is taken for studying coalition formation, stability analysis, bargaining or profit allocation [Nagarajan and Sosic, 2008]. The non-cooperative models on the other hand, usually apply sequential games; especially the use of the *principal – agent* model of contracting theory [Laffont, 2001; Salanié, 2005] is common. In the operational research literature this approach is called *supply chain coordination* [Arshinder *et al.*, 2008]. These researches are usually related to inventory management in distributed production planning problems. Most of the works study the distributed version of the one-period *newsvendor lot-sizing problem* due to its simple structure; for a review of newsvedor games we refer to [Cachon and Netessine, 2004].

Two main problems with the majority of the current studies in the literature are that they (i) consider rather special production problems lacking generality, and (ii) usually take only simple concepts from game theory (like e.g., the Stackelberg games [Hennet and Arda, 2008]). Both the supply chain management and the game theory literature have some practically more relevant results which should be combined and further studied, such as rolling horizon planning, hierarchical planning systems, repeated games, equilibrium learning, Vickrey – Clarke – Groves mechanisms, to name a few. In the mechanism design theory there are also recent achievements considering algorithmic issues, such as verification ([Nisan and Ronen, 2001]), distributed mechanisms ([Shneidman and Parkes, 2004]) and stochastic problems ([Ieong *et al.*, 2007; Papakonstantinou *et al.*, 2011]). This paper intends to be a further step on this way.

The remainder of the paper is organised as follows. In Section 3, we model distributed decision making in the supply network as a mechanism design problem and inspect some of its properties. We introduce some assumptions into the model in Section 4, in order to develop practically applicable efficient coordination mechanisms. In Section 5, we illustrate the performance of the VMI compared to the traditional order-based purchase on a numerical example. Finally, we conclude the results of this paper and suggest some future research directions.

3 A Mechanism Design Analysis

In this section we formalise the supply network model with n retailer agents and a supplier agent. For the sake of simplicity, we assume that the *retailers* are homogeneous, although this assumption can be relaxed and the results still remain valid. Retailer i has some private belief (forecast) about the future market *demand*, which is denoted by $\theta_i \in \Theta$. Demand is satisfied by production done at the *supplier* who has, in turn, private information about the cost factors. Since the exact demands $\xi_i \in D$ realise at some later time, only the forecasts can be considered when creating a *production plan* denoted by $x \in \mathcal{K}$. If the actual demand does not match the forecast, then the production will deviate from what was planned. If the demand was underestimated, new and costly production is necessary, while overestimation leads to extra, sometimes even to obsolete inventories. In both cases, the actual costs incurred are higher than planned. Therefore the production cost at the supplier is a function of the original production plan as well as of the realized demands: $c \in C = \{\, c : \mathcal{K} \times D^n \to \mathbb{R} \,\}$, which is a private information of the supplier. Note that we do not assume that an a priori distribution about the private information is known by the other agents, i.e., we regard a situation with *strict incomplete information*.

According to the classic mechanism design theory, an independent mediator, the *mechanism* is required for observing the agents' actions, making the decision and after realisation, transferring the payments among the agents. Some recent developments aim at omitting the mediator, which possibility we also will study in the next section. For the moment, let us define the mechanism as $\mathcal{M} = (f, t_1, \ldots, t_n, t_s)$, where $f : \Theta^n \times C \to \mathcal{K}$ is the choice function determining the production plan based on the forecasts and the cost function[1], $t_i : \Theta^n \times C \times D^n \to \mathbb{R}$ are the payment functions of the retailers ($i = 1, \ldots, n$), and $t_s : \Theta^n \times C \times D^n \to \mathbb{R}$ is the payment for the supplier.

Note two assumptions of this formulation. Firstly, this is a *direct-revelation* mechanism, i.e., the strategy of the agents is to share their private information (not necessarily truthfully) with the mediator. Secondly, we consider that the realised demands are commonly observable.

After the demands realise, an income arises at each retailer i from the sales: $v_i : D \to \mathbb{R}$. Now we can define the utility—the income minus the cost—for each retailer and the supplier:

$$u_i(\theta_i, \hat{\theta}, \hat{c}, \xi) = v_i(\xi_i) - t_i(\hat{\theta}, \hat{c}, \xi) \tag{1}$$

and

$$u_s(c, \hat{\theta}, \hat{c}, \xi) = t_s(\hat{\theta}, \hat{c}, \xi) - c(f(\hat{\theta}, \hat{c}), \xi) \tag{2}$$

respectively, if their private information is θ_i and c, but they claim $\hat{\theta}_i$ and $\hat{c}$ instead, with $\theta = (\theta_1, \ldots, \theta_n)$ and $\xi = (\xi_1, \ldots, \xi_n)$ denoting the forecast and demand vectors. (The

[1] Although the production planning problem is complex in general, in this paper we disregard computational issues, and assume that an optimal plan can be found for every possible forecast.

first parameters of the utility functions are the real private information, the second and third are the communicated parameters, and the last one is the realised demand.)

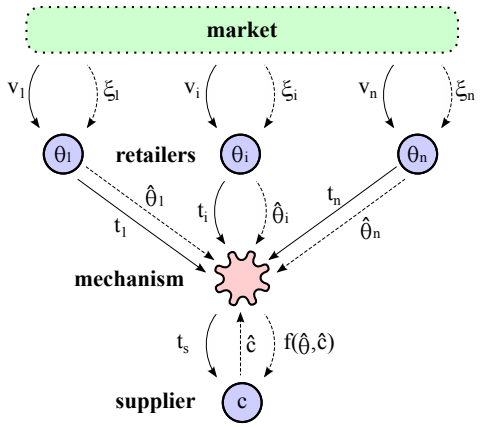


Figure 1: Mechanism design setting.

Figure 1 illustrates this mechanism design model, where the dashed arrows denote information flow, while the solid ones represent the monetary payments. The variables and functions inside the agents are private knowledge of the given agent and are unobservable for the others.

We are seeking such a mechanism, wherewith the performance of the production network as a whole is optimal. This can be guaranteed, if all the agents disclose their private information truthfully, and the mechanism uses an optimal planning choice function. Let us define these properties formally.

Definition 1 *A mechanism $\mathcal{M}$ is* (weakly) strategy-proof*, if truth telling is a* dominant strategy *for every agent, i.e., it maximizes their expected utility: $\forall i, \forall \theta_i \in \Theta, \forall \hat{\theta} \in \Theta^n, \forall \hat{c} \in C$:*

$$\mathbb{E}_{\tilde{\theta}}[u_i(\theta_i, \tilde{\theta}, \hat{c}, \xi)] \geq \mathbb{E}_{\tilde{\theta}}[u_i(\theta_i, \hat{\theta}, \hat{c}, \xi)], \tag{3}$$

where $\tilde{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_{i-1}, \theta_i, \hat{\theta}_{i+1}, \ldots, \hat{\theta}_n)$, and $\forall c, \hat{c} \in C, \forall \hat{\theta} \in \Theta^n$:

$$\mathbb{E}_{\hat{\theta}}[u_s(c, \hat{\theta}, c, \xi)] \geq \mathbb{E}_{\hat{\theta}}[u_s(c, \hat{\theta}, \hat{c}, \xi)]. \tag{4}$$

Definition 2 *The choice function f is* efficient*, if it maximises* social welfare *(the sum of the utilities without the payments), i.e., $\forall \theta \in \Theta^n, \forall c \in C$:*

$$f(\theta, c) \in \operatorname*{argmax}_{x \in \mathcal{K}} \mathbb{E}_\theta[\sum v_i(\xi_i) - c(x, \xi)] = \operatorname*{argmin}_{x \in \mathcal{K}} \mathbb{E}_\theta[c(x, \xi)]. \tag{5}$$

Firstly, we show that if a strategy-proof mechanism gives the same output for different cost functions, then it is expected to give the same payment to the supplier.

Proposition 1 *If $\mathcal{M}$ is a strategy-proof mechanism, $c, \hat{c} \in C, \theta \in \Theta^n$ such that $f(\theta, c) = f(\theta, \hat{c})$, then $\mathbb{E}_\theta[t_s(\theta, c, \xi)] = \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)]$.*

Proof Let us assume that $\mathbb{E}_\theta[t_s(\theta, c, \xi)] < \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)]$ (the other direction is analogous). But then

$$\mathbb{E}_\theta[t_s(\theta, c, \xi)] - \mathbb{E}_\theta[c(f(\theta, c), \xi)] < \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)] - \mathbb{E}_\theta[c(f(\theta, \hat{c}), \xi)], \tag{6}$$

i.e., the mechanism is not strategy-proof, since the supplier with cost function c would reveal $\hat{c}$ instead. □

We now prove that if we are looking for an efficient strategy-proof mechanism, then t_s should be (in the sense of expected value) independent from the cost function of the supplier, therefore it excludes the possibility of cost sharing among the agents.

Theorem 2 *Let $\mathcal{M}$ be an efficient, strategy-proof mechanism. Then*

$$\forall \theta \in \Theta^n, \forall c, \hat{c} \in C : \mathbb{E}_\theta[t_s(\theta, c, \xi)] = \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)]. \tag{7}$$

Proof The proof is similar to the proof of the uniqueness of *Groves mechanism* among efficient and strategy-proof mechanisms proved by [Green and Laffont, 1977], thus we exploit that the cost function can be arbitrary.

Let us consider a fixed θ, and indirectly assume that the statement of the theorem is false: $\exists c, \hat{c} \in C$: $\mathbb{E}_\theta[t_s(\theta, c, \xi)] > \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)]$. Furthermore, let us define

$$\varepsilon = \mathbb{E}_\theta[t_s(\theta, c, \xi)] - \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)] > 0. \tag{8}$$

Due to the *modus tollens* of Proposition 1, $f(\theta, c) \neq f(\theta, \hat{c})$. Because the cost function can be arbitrary, $\exists \tilde{c} \in C, \exists k \in \mathbb{R}$:

$$\mathbb{E}_\theta[\tilde{c}(f(\theta, \hat{c}), \xi)] = k \tag{9}$$

$$\mathbb{E}_\theta[\tilde{c}(x, \xi)] > k \qquad \forall x \neq f(\theta, \hat{c}) \tag{10}$$

$$\mathbb{E}_\theta[\tilde{c}(f(\theta, c), \xi)] < k + \varepsilon. \tag{11}$$

From the efficiency of the mechanism follows that $f(\theta, \tilde{c}) = f(\theta, \hat{c})$, and then from Proposition 1, we have $\mathbb{E}_\theta[t_s(\theta, \tilde{c}, \xi)] = \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)]$. But then

$$\mathbb{E}_\theta[u_s(\tilde{c}, \theta, \tilde{c}, \xi)] = \mathbb{E}_\theta[t_s(\theta, \tilde{c}, \xi)] - \mathbb{E}_\theta[\tilde{c}(f(\theta, \tilde{c}), \xi)] = \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)] - k, \tag{12}$$

and

$$\mathbb{E}_\theta[u_s(\tilde{c}, \theta, c, \xi)] = \mathbb{E}_\theta[t_s(\theta, c, \xi)] - \mathbb{E}_\theta[\tilde{c}(f(\theta, c), \xi)] > \mathbb{E}_\theta[t_s(\theta, c, \xi)] - k - \varepsilon = \mathbb{E}_\theta[t_s(\theta, \hat{c}, \xi)] - k, \tag{13}$$

thus the mechanism is not strategy-proof. □

The theorem proves the reasonable conjecture that the supplier can claim higher costs in such a way, that the optimal production plan does not change. Thus, the supplier may try to obtain more payment without increasing its costs.

4 Coordination Mechanisms for Supply Networks

Although Theorem 2 is rather negative if we are aimed at fair cost sharing, it has some positive consequences as well that

we analyse in this section. Our main goal is to omit the necessity of an independent mediator which is unrealistic in a supply network, but at the same time, to preserve the favourable properties of the system. In what follows, we dissolve the two reasons for the existence of an independent mechanism: balancing the difference between the agents' payments and providing efficiency.

From now on, we assume that the payment of the supplier is independent from its revealed cost function—explained by the conclusions of Section 3—, and the next definition necessitates that the payments of the retailers are also independent from $\hat{c}$.

Definition 3 *A mechanism $\mathcal{M}$ is* budget-balanced*, if*

$$\forall \hat{\theta} \in \Theta^n, \forall \xi \in D^n : t_s(\hat{\theta}, \xi) = \sum t_i(\hat{\theta}, \xi), \qquad (14)$$

i.e., there is no surplus or deficit for the mechanism, the total payment is distributed among the agents.

Note that requiring budget-balance excludes the application of the Vickrey – Clarke – Groves (VCG) mechanisms which is one of the main positive results of the classic mechanism design theory, and it is also frequently applied for solving algorithmic problems [Nisan *et al.*, 2007].

A direct corollary of the independence of t_s from $\hat{c}$ is that the supplier agent can maximise its utility by minimising its cost. This means that the efficiency of the mechanism corresponds with the supplier's interest, therefore the mechanism can be implemented by the supplier without requiring an independent mediator. There is no need to disclose its private information about the costs, and furthermore, even the specific planning algorithm and the resulted plan can be kept secret. In fact, this property is the essence of VMI.

In order to provide strategy-proofness, truth telling should be the dominant strategy for each retailer, independently from the decision and realised demand of the other retailers. Since the income is independent from the disclosed information, the utility is maximal when the payment is minimal.

Firstly, let us consider a trivial example for illustration, when the forecast is simply the expected value of the demand. In this case it is easy to see that for example the payment function in the form

$$t_i(\hat{\theta}, \xi) = \alpha_i |\hat{\theta}_i - \xi_i| + \beta_i(\hat{\theta}_{-i}, \xi), \qquad (15)$$

where $\alpha_i > 0$ is a constant, β_i is an arbitrary function and $\hat{\theta}_{-i} = (\hat{\theta}_1, \ldots, \hat{\theta}_{i-1}, \hat{\theta}_{i+1}, \ldots, \hat{\theta}_n)$, is appropriate, since the first term is expected to be minimal when $\hat{\theta}_i = \mathbb{E}[\xi_i]$, and the second term is independent from $\hat{\theta}_i$.

If $\beta_i(\hat{\theta}_{-i}, \xi)$ depends on $\hat{\theta}_j$ and ξ_j ($j \neq i$), this allows some profit sharing between the retailers, but requires cooperation between them. Otherwise $\beta_i(\hat{\theta}_{-i}, \xi) = \gamma_i(\xi_i)$ with some arbitrary γ_i function, and the retailers' profits are independent from each other. In this case, $\gamma_i(\xi_i)$ practically can be considered as the payment for the supplied products, while α_i defines the price of the flexible VMI service.

When the forecast becomes more complex, it is not straightforward to guarantee strategy-proofness. If for example we refine the previous model assuming the forecast is given by the *expected value* and the *standard deviation*, the task becomes more interesting. The difference between the expected and realised demand can be easily measured, but how can we estimate the accuracy of the standard deviation based only on one observation? In the following, we answer this question by presenting a strongly strategy-proof payment, which is a generalisation of the result published in [Egri, 2008], without assuming any particular distribution of the demand. Due to the lack of space, we omit the proof which is analogous to the one presented in the previously mentioned thesis.

Theorem 3 *Let us consider a one-period supply coordination network problem, where the forecasts are given by the expected values and the standard deviations, i.e., $\theta_i = (m_i, \sigma_i)$. Then the payment function in the form*

$$t_i(\hat{m}, \hat{\sigma}, \xi) = \alpha_i \left(\frac{(\hat{m}_i - \xi_i)^2}{\hat{\sigma}_i} + \hat{\sigma}_i \right) + \beta_i(\hat{m}_{-i}, \hat{\sigma}_{-i}, \xi), \qquad (16)$$

where $\alpha_i > 0$ is a constant and β_i is an arbitrary function, is strongly strategy-proof.

One can notice the similarity between this payment and the payment defined by Eq. (15). Furthermore, there is a simple intuition behind the term $(\hat{m}_i - \xi_i)^2/\hat{\sigma}_i + \hat{\sigma}_i$: if a retailer states that the forecast is fairly precise (i.e., σ_i is small), it is ready to pay larger compensation for the difference between the expected and the realised demand. This could be avoided by stating higher uncertainty, but then this increases the second part of the term.

Such one-period problems are widely studied in the supply chain coordination literature due to their simple structure, however, in several practical cases they cannot be properly applied. In industrial problems involving longer horizons, usually some medium-term, discrete forecasts are used; in addition, the forecasts are often updated from time to time, on a rolling horizon. These more realistic cases can be approached in a similar way as the one-period problem: besides the payment for the products (the "β-part" of the payment), the *imprecision of the forecast* should be measured and used as the basis for the payment of the VMI service ("α-part"). For example, in [Váncza *et al.*, 2008] we present a strongly strategy-proof payment scheme for the *multi-period*, *rolling horizon* case with uncertain length of product life-cycle.

All in all, with VMI the supplier not only offers products, but also flexibility as a service. Accordingly, a composite payment function should be constructed: the retailers must pay not only (i) for the quantity delivered, but also (ii) for the deviation from the forecast, as well as (iii) for the uncertainty of the forecast. This payment compensates the supplier for the eventual obsolete inventory or the cost of extra production exceeding its original production plan.

5 Computational Study

In this section we illustrate on a simple example how an efficient strategy-proof mechanism can improve the performance of a supply network. We consider n retailers, and we assume that the ξ_i demands are independent and normally distributed with expected values m_i and standard deviations σ_i.

Firstly, we examine a suboptimal solution, where the retailers make firm orders $\hat{m}_i$ and pay a w_1 wholesale price for the supplied goods. When the ξ_i demand realises, either some surplus remain at retailer i, or it has to order again, but due to the urgency, on a higher w_2 price. The supplier works in *make-to-order* mode, i.e., it produces the normal orders with c_1 piecewise cost, and the urgent orders on a higher c_2 cost. Formally, this can be expressed as follows:

$$t_i(\hat{m}, \hat{\sigma}, \xi) = w_1 \hat{m}_i + w_2 \max(\xi_i - \hat{m}_i, 0), \qquad (17)$$

and the emerging cost at the supplier becomes

$$c(\xi) = c_1 \sum \hat{m}_i + c_2 \sum \max(\xi_i - \hat{m}_i, 0). \qquad (18)$$

One can derive that in this newsvendor-like case the optimal order quantity is

$$\hat{m}_i = F_i^{-1}(1 - \frac{w_1}{w_2}), \qquad (19)$$

where F_i is the cumulative density function (CDF) of ξ_i, thus the optimal order quantity is not even equal with the expected demand.

However, if one applies a mechanism with a payment defined by Eq. (16), then the retailers truthfully reveal their private information about the expected values and standard deviations of the demand forecasts. Now, the expected value of the total demand will be the sum of the expected values, and since the demands are considered to be independent, the standard deviation of the total demand becomes $\sqrt{\sum \sigma_i^2}$. Furthermore, if the demand at the retailers are normally distributed, the distribution of the total demand will also be normal, therefore the optimal production quantity and cost are

$$x = F^{-1}(1 - \frac{c_1}{c_2}) \qquad (20)$$

and

$$c(x, \xi) = c_1 x + c_2 \max(\sum \xi_i - x, 0), \qquad (21)$$

where F is the CDF of the total demand.

Figure 2 illustrates the difference between the costs of the two approaches, depending on the number of retailers. We set the price parameters as $c_1 = 50$, $c_2 = 80$, $w_1 = 135$, and $w_2 = 165$. The expected value and standard deviation of the total demand was set to $m = 800$ and $\sigma = 100$, and we considered retailers with identical distributions, therefore their parameters were $m_i = m/n$ and $\sigma_i = \sigma/\sqrt{n}$. Each cost value indicated on the figure is an average made on 5000 simulation runs.

As it can be seen, the coordinated VMI is more efficient than the order-based supply even in the one retailer case due to the elimination of the double marginalisation. However, when the number of the retailers increases, risk pooling keeps the optimality in the network, while the uncoordinated approach quickly deviates from cost efficient performance.

6 Conclusions and Further Work

In this paper we studied networks of autonomous retailers and a supplier, where the utilities depend on a stochastic market demand whose distribution is not known by the decision

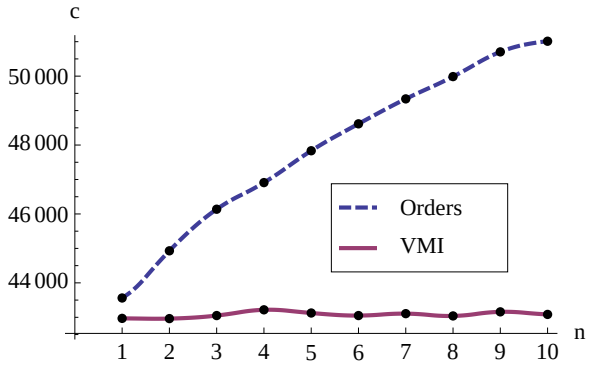


Figure 2: The cost in function of the number of retailer agents.

maker. The basic assumptions of our model are rooted in supply chain coordination models, specifically, we departed from the vendor managed inventory where demand anticipated and communicated by the retailer drives production planning at the supplier. The analysis was aimed at elaborating a direct revelation mechanism that maximizes overall utility, even though the partners are rational, expected utility maximising decision makers. The coordination problem called also for a budget-balancing solution. Furthermore, we were interested in finding an answer to a crucial question in supply chain coordination: the sharing of costs and profits. Due to the private information of the players, the traditional distribution methods of cooperative game theory were not applicable.

Firstly, we formulated a mechanism design model with the inherent incomplete information about the forecasted demand and production costs. We proved that in any mechanism that is efficient and where truth-telling is a dominant strategy for each player, the payment for the supplier's production efforts should be independent from its actual cost. Hence, this result excludes in general the possibility of sharing costs in a fair way among the agents.

Next, we presented a specific coordination mechanism that was efficient, budget-balanced and strategy-proof. The managerial insight behind these results is that VMI should be interpreted as a *service* provided by the supplier. In turn, the retailer's payment for this service should depend on the accuracy of the information it shares with the supplier. For the practical applicability, the mechanism requires (i) no new mediator party, and (ii) no new information exchange channels, however, it implies that (i) new innovative business models, (ii) efficient local planning at the supplier, and (iii) improved forecasting of market demands are essential.

Negative consequences of Theorem 2 are not entirely discouraging, though. By relaxing assumptions of our model in various ways, a number of open questions emerge, together with new opportunities for the application of mechanism design in supply chain management. In this paper we assumed that the entire demand should be fulfilled. One can study the situation when *lost sales* are allowed, in which case Theorem 2 holds no more. Furthermore, one may also study such mechanisms that are only approximately efficient.

Of course, not only the VMI supply scheme is important; the traditional order-based procurement can also be analysed by means of the mechanism design theory. In such cases negotiation processes—possibly in an automated way between enterprise information systems—could be developed for supporting decentralised decision making. This approach, however, cannot skip considering computational issues any more [Nisan and Ronen, 2001]. We made a first step in this direction in [Egri *et al.*, 2011], where we presented a simple protocol aimed at improving benefits both for a manufacturer and its supplier, without guaranteeing strict optimum.

Finally, a further goal is extending the two-echelon games and handling more complete supply networks. In such cases, achieving global optima through coordination is out of question due to the complexity of the interconnections, the local planning problems to be solved, as well as the dynamic behaviour of the network. Nevertheless, the analytic approach can help estimate the theoretic bounds of the system, and measure the performance gap between results of approximate optimisations and the theoretical global optimum.

Acknowledgements

This work has been supported by the Hungarian Scientific Research Fund OTKA No. T73376 and the National Office for Research and Technology OMFB No. 01638/2009 grants.

References

[Arshinder *et al.*, 2008] Arshinder, A. Kanda, and S.G. Deshmukh. Supply chain coordination: Perspectives, empirical studies and research directions. *International Journal of Production Economics*, 115(2):316–335, 2008.

[Cachon and Netessine, 2004] G.P. Cachon and S. Netessine. Game theory in supply chain analysis. In D. Simchi-Levi, S.D. Wu, and Z-J. Shen, editors, *Handbook of Quantitative Supply Chain Analysis: Modeling in the eBusiness Era*, pages 13–66. Kluwer, 2004.

[Dash *et al.*, 2003] R.K. Dash, N.R. Jennings, and D.C. Parkes. Computational mechanism design: A call to arms. *IEEE Intelligent Systems*, 18:40–47, 2003.

[Egri and Váncza, 2007] P. Egri and J. Váncza. Cooperative production networks – multiagent modeling and planning. *Acta Cybernetica*, 18(2):223–238, 2007.

[Egri *et al.*, 2011] P. Egri, A. Döring, T. Timm, and J. Váncza. Collaborative planning with benefit balancing in Dynamic Supply Loops. *CIRP Journal of Manufacturing Science and Technology*, 2011. In print, doi: 10.1016/j.cirpj.2011.05.002.

[Egri, 2008] P. Egri. *Coordination in Production Networks*. PhD thesis, Eötvös Loránd University, Budapest, 2008.

[Green and Laffont, 1977] J. Green and J-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45(2):427–438, 1977.

[Hennet and Arda, 2008] J-C. Hennet and Y. Arda. Supply chain coordination: A game-theory approach. *Engineering Applications of Artificial Intelligence*, 21:399–405, 2008.

[Ieong *et al.*, 2007] S. Ieong, A. Man-Cho So, and M. Sundararajan. Stochastic mechanism design. In *Proceedings of the 3rd International Conference on Internet and Network Economics*, pages 269–280, 2007.

[Laffont, 2001] D. Laffont, J-J.and Martimort. *The Theory of Incentives : The Principal-Agent Model*. Princeton University Press, 2001.

[Monostori *et al.*, 2006] L. Monostori, J. Váncza, and S.R.T. Kumara. Agent-based systems for manufacturing. *CIRP Annals–Manufacturing Technology*, 55(2):697–720, 2006.

[Nagarajan and Sosic, 2008] M. Nagarajan and G. Sosic. Game-theoretic analysis of cooperation among supply chain agents: Review and extensions. *European Journal of Operational Research*, 187(3):719–745, 2008.

[Nisan and Ronen, 2001] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.

[Nisan *et al.*, 2007] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[Papakonstantinou *et al.*, 2011] A. Papakonstantinou, A. Rogers, E.H. Gerding, and N.R. Jennings. Mechanism design for the truthful elicitation of costly probabilistic estimates in distributed information systems. *Artificial Intelligence*, 175(2):648–672, 2011.

[Rosenschein and Zlotkin, 1994] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, Cambridge, MA, USA, 1994.

[Salanié, 2005] B. Salanié. *The Economics of Contracts: A Primer, 2nd Edition*. The MIT Press, 2005.

[Shneidman and Parkes, 2004] J. Shneidman and D.C. Parkes. Specification faithfulness in networks with rational nodes. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing*, pages 88–97, 2004.

[Shoham and Leyton-Brown, 2008] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

[Simchi-Levi *et al.*, 2000] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and Managing the Supply Chain: Concepts, Strategies, and Cases*. McGraw – Hill, New York, 2000.

[van der Krogt *et al.*, 2008] R. van der Krogt, M. de Weerdt, and Y. Zhang. Of mechanism design and multiagent planning. In *Proceeding of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 423–427, 2008.

[Váncza and Márkus, 2000] J. Váncza and A. Márkus. An agent model for incentive-based production scheduling. *Computers in Industry*, 43:173–187, 2000.

[Váncza *et al.*, 2008] J. Váncza, P. Egri, and L. Monostori. A coordination mechanism for rolling horizon planning in supply networks. *CIRP Annals – Manufacturing Technology*, 57(1):455–458, 2008.

Multi-Agent Based Collaborative Demand and Capacity Network Planning in Heterarchical Supply Chains

Bernd Hellingrath and Peer Küppers
Chair for Information Systems and Supply Chain Management
WWU Münster
Leonardo-Campus 3, 48149 Münster, Germany
{Bernd.Hellingrath, Peer.Kueppers}@ercis.uni-muenster.de

Abstract

In the light of a high division of labor, decentralized coordination shows major importance in today's production and logistics networks and hence is subject to current supply chain management (SCM) research. Especially collaborative planning concepts promise to meet the specific requirements imposed by heterarchical supply chains. This paper demonstrates the multi-agent based modeling and information system-implementation of one concrete collaborative planning concept, namely the collaborative demand and capacity network planning approach. The paper's intention is to substantiate the suitability of multi-agent concepts in decentralized SCM and to strengthen the motivation to combine multi-agent and SCM research.

1 Motivation and Problem Description

Today, producing companies are facing a highly dynamic competitive environment which requires effective and efficient business processes in order to meet customer needs adequately. Measures like the concentration on core competencies aiming at this issue have led to a high division of labor and hence a growing number of legally independent participants in the process of providing products to the ultimate customer. The resulting inter-organizational network – commonly denoted as supply chain (SC) – consists of several autonomous companies which cannot necessarily be forced to follow decisions or plans of a superordinate unit. Thus, coordination in such SCs cannot be achieved in the same way as in hierarchical organizations which can be controlled by one dominant actor.

Coordination mechanisms for *hierarchical* SCs have been researched intensely, resulting in the development and practical application of sophisticated supply chain management (SCM) methods, e.g. implemented in advanced planning systems. However, these hierarchical coordination approaches cannot be applied to heterarchical SCs due to the aforementioned decision autonomy of SC actors, combined with the unwillingness to share sensitive private information. Furthermore, companies are generally involved in several networks with conflicting claims, e.g. on a company's resources, which makes central hierarchical plans inappropriate (for details see for example [Breiter *et al.*, 2009]).

Several approaches in current SCM research address the aspects and peculiarities of *heterarchical* SCs in the development of adequate coordination mechanisms (for an overview cf. [Breiter *et al.*, 2009], [Stadtler, 2009]). Especially approaches in the domain of collaborative planning (CP) promise to meet the requirements imposed by heterarchical SCs with respect to decision autonomy and privacy of information. These coordination mechanisms intend to overcome the restrictions of traditional hierarchical planning concepts regarding the practical applicability in heterarchical SCs, while simultaneously improving the SC's performance compared to the uncoordinated situation being present in successive planning approaches.

With respect to the representation and information system (IS) based implementation of suchlike decentralized coordination mechanisms, multi-agent systems (MAS) promise to meet the specific requirements induced by heterarchical structures (cf. [Breiter *et al.*, 2009]). Heterarchical SCs consist of several autonomous actors following their own objectives while being interdependent and linked by physical, financial and informational flows. Thus, interactions between autonomous companies are a major constituent in such networks. MAS provide a natural metaphor for the representation of suchlike complex systems (cf. [Moyaux *et al.*, 2006]) and a technological basis to handle interactions between the SC actors. The suitability of MAS in SCM is, for example, discussed in [Moyaux *et al.*, 2006]. Consequently, MAS provide an elaborate means to represent and implement CP approaches in IS.

This paper presents one concrete multi-agent based modeling and IS-implementation of a CP coordination concept, the collaborative Demand and Capacity Network Planning (DCNP) (cf. [Hellingrath and Hegmanns, 2010]). This approach is suitable for coordinating demanded and offered capacities in built-to-order (BTO) production and logistics networks as existent e.g. in the automotive industry. These SCs are characterized by a huge number of configurable product variants and hence large parts of these SCs – spreading the planning domains of several autonomous companies – follow a BTO production strategy.

On the one hand, the intention of this paper is to demonstrate and substantiate the suitability of MAS concepts in

the context of SCM by applying them to the DCNP approach. On the other hand, the presented results are embedded into a more general research on an MAS-based framework for modeling and evaluating arbitrary CP mechanisms called the Framework for Intelligent Supply Chain Operations (FRISCO, cf. [Hellingrath *et al.*, 2009]). The described MAS-based modeling and evaluation of the collaborative DCNP concept therefore constitutes a demonstration of parts of the research on FRISCO.

The paper is structured as follows. Chapter 2 provides an introduction to the collaborative DCNP coordination mechanism. Here, the general idea, intra-organizational planning and inter-organizational interaction and negotiation processes are presented. In chapter 3 the MAS-based modeling of this CP concept is presented. A short introduction to the FRISCO framework (especially its MAS-based modeling component) is provided and the respective MAS-models of the collaborative DCNP are described. Chapter 4 provides details on the proof-of-concept implementation of this coordination mechanism. The implementation is based on the developed MAS-models, i.e. a model-driven development (MDD) approach is followed. The paper concludes with chapter 5, providing an evaluation of the development results and giving an outlook on future research to be conducted in this context.

2 Collaborative Demand and Capacity Network Planning

The performance of today's SCs is not determined by a single enterprise's activities, but relies on the whole network's effectiveness and efficiency due to the high division of labor. Current practices show that long-term contracts specify relatively fixed performance agreements between the network's actors, e.g. with respect to the provision of capacity (cf. [Schuh, 2006]). These agreements show a certain inflexibility in the case of demand variations, leading to unsatisfied demand and/or inefficiencies in the production processes due to capacity under-utilization. These aspects are a major issue in BTO production and logistics networks which simultaneously require on time demand fulfillment as well as cost efficient production processes. In case of demand variations, an over- or under-utilization of contractually agreed capacities is likely to occur, leading to late deliveries or inefficient capacity utilization.

The collaborative DCNP concept aims at solving this issue by providing mechanisms to improve the match between demanded and required capacity in inter-organizational BTO production and logistics networks. Three kinds of SC actors that can participate in the coordination mechanism are distinguished: build-to-stock (BTS) suppliers constitute the upstream "ends", original equipment manufacturers (OEM) the downstream "ends" of the coordinated SC. BTO suppliers are intermediaries which source from BTS (or BTO) suppliers and supply OEMs (or other BTO suppliers). The concept is based on the idea of CP and hence coordination is achieved by mutual agreement between these actors without the revelation of private information and loss of local decision autonomy.

On the one hand, the planning concept requires performance agreements between companies which specify the planned capacity provided from a supplier to its customer in the mid-term planning horizon. On the other hand, the concept defines procedures that automatically identify and solve conflict situations, i.e. detect capacity shortages and help to overcome them by a decentralized adaptation of network-wide plans (cf. Figure 1). Demand for capacity is monitored at each stage for each supplier and compared to the previously specified performance agreements in order to be able to identify violations of the agreed capacity corridors.

Besides monitoring the performance agreements, the collaborative DCNP concept defines inter-organizational procedures to resolve occurring contract violations. These procedures are intended to be performed automatically in order to achieve a network-wide, mutually agreed adaptation of capacities to the current demand situation. The concept achieves this decentralized coordination via bilateral negotiations between the actors on multiple stages of the network. The automated negotiations contain the exchange of requests and responses regarding adaptations of plans, i.e. performance agreements, and invoke several local planning processes in order to determine the adaptions' effects on an actor's local plans. Compensation payments for the reservation of additional or cancellation of unnecessary capacities provide a means for the acceptance of locally unfavorable plans. Thus, incentives for participation are integrated into this coordination mechanism, i.e. globally (SC-wide) preferable plans can be achieved in the collaborative DCNP concept.

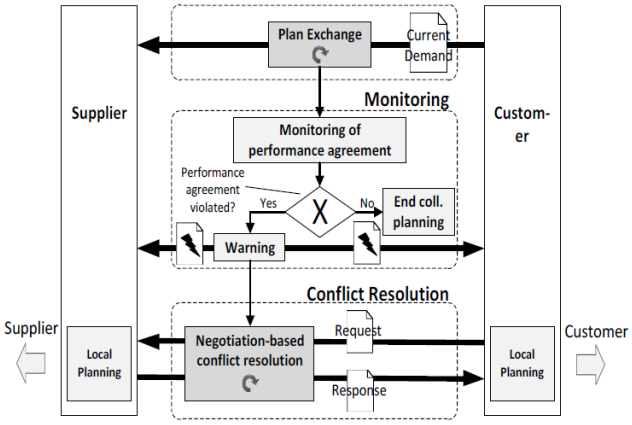


Figure 1: Inter-organizational planning in the collaborative DCNP concept (cf. [Hellingrath and Hegmanns, 2010])

The inter-organizational planning processes adjust the bilateral performance agreements continuously to the current demand situation by solving the aforementioned capacity agreement violations. This coordination process involves multiple rounds of planning and negotiating in order to achieve a network-wide feasible and agreed plan. The high-level description of the collaborative DCNP intra- and inter-organizational processes is depicted in Figure 1.

As discussed in chapter 1, MAS provide the concepts to represent the heterarchical network structure, the interaction processes and the information flows between the different SC participants in the described collaborative DCNP approach. Furthermore, they provide a technological basis to implement the coordination concept in an IS. Thus, the next chapter describes a conceptual model of the collaborative DCNP in a multi-agent based modeling language. Following the concept of MDD, this model provides the basis for the IS-implementation of the collaborative DCNP approach which is presented in chapter 4.

3 Multi-Agent Based Model of the Collaborative DCNP Concept

The collaborative DCNP concept shows the characteristics of CP and hence it provides a reference that can be used to exemplary evaluate the FRISCO framework into which the results of this paper are embedded. The general idea of FRISCO is to provide an environment that allows modeling and evaluating decentralized coordination mechanisms – especially CP concepts – for arbitrary heterarchical SCs. The ultimate goal of this research is to pave the way for suchlike coordination mechanisms from primarily being a research domain to practical applications in real SCs. Basically, the framework consists of two parts: a *modeling* and an *evaluation* environment for CP coordination mechanisms. Due to the aforementioned suitability for heterarchical SCs and CP, FRISCO is based on MAS concepts. In order to be able to efficiently cope with differently shaped CP mechanisms beyond the mere modeling, an approach in analogy to MDD was chosen. The goal is to provide an environment that allows modeling the complex structures and processes of CP concepts and furthermore to use these models in order to automatically create executable code for an evaluation of the CP approaches in different scenarios.

Modeling MAS and the automated transformation of models to executable code has been researched intensely in the MAS context and several approaches have evolved (for an overview see e.g. [Nunes *et al.*, 2009]). For the proof-of-concept implementation of the collaborative DCNP coordination approach, the DSML4MAS domain specific modeling language (cf. [Hahn *et al.*, 2009]) was chosen. This language provides the required concepts for a representation and implementation of CP approaches, especially with respect to the graphical definition of complex interaction protocols required by CP and especially the collaborative DCNP approaches.

The abstract syntax of this modeling language is described by an agent-platform independent metamodel called PIM4Agents (cf. [Hahn *et al.*, 2009]). The concrete syntax of this modeling language has been specified and implemented by means of the Eclipse Modeling Framework (EMF), i.e. graphical modeling of complex MAS is supported by the DSML4MAS environment. Furthermore, translation rules have been defined transforming conceptual models to executable code which can be run on the FIPA compliant MAS platform JADE.

The DSML4MAS relies on several views on the different aspects of MAS (for details see [Hahn *et al.*, 2009]). By means of these views, the DSML4MAS development environment provides all concepts that are required to formally model the collaborative DCNP approach. All required views were developed and instantiated in order to model the concept. The most important models that describe the CP concept are depicted in the following.

As mentioned in chapter 2, collaborative DCNP distinguishes three types of actors (BTS/BTO suppliers and OEM) that can participate in the coordination mechanism. Transferred to the MAS model in DSML4MAS, these are represented by three agents in the so called "agent view" (cf. Figure 2). The collaborative DCNP coordination mechanism relies on two roles that can be taken on by its participants: *Supplier* and *Customer*. These roles are reflected in the agent view by permitting the agent types to act in the respective role. The OEM determines the downstream "end" of the production network and is therefore only permitted to act as a *Customer*. Analogously, upstream the BTS supplier constitutes the last actor in the coordinated part of the SC, i.e. is only permitted to be a *Supplier*. The multi-tier support of the collaborative DCNP concept is facilitated by allowing BTO suppliers to act as both, *Suppliers* and *Customers*. Besides the assignments between agents and roles, the agent view is furthermore used to define plans that can be used by the agents, i.e. it allows for modeling the internal agent behavior. Plans that are directly connected to an agent via a *uses* relation are triggered in the instantiation phase. Thus, the agent diagram depicted in Figure 2 shows the startup plans that initialize the agents' data structures. Besides these three plans, the agent diagram contains a multitude of plans that describe the agents' local behavior throughout the collaborative DCNP concept (not depicted in Figure 2).

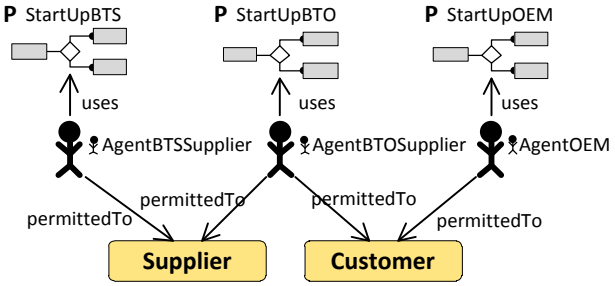


Figure 2: Agent view on the collaborative DCNP concept (excerpt)

The next step in modeling the collaborative DCNP concept is the definition of interactions that can take place in the MAS, i.e. in this case between the *Customer* and *Supplier* roles. This requires the assignment of the two roles to an organizational structure, which is modeled by an organization called *OrganizationDCNP* in the "MAS view" (see Figure 3, left). The *OrganizationDCNP* represents one stage of the SC being coordinated by the collaborative DCNP concept. Modeling the organization is the prerequisite for the definition of interaction mechanisms taking place in the organization as the participating roles are defined. Besides

the organizational assignment, the MAS view also contains a definition of the environment (not depicted in Figure 3), especially containing descriptions of the messages being communicated in the MAS.

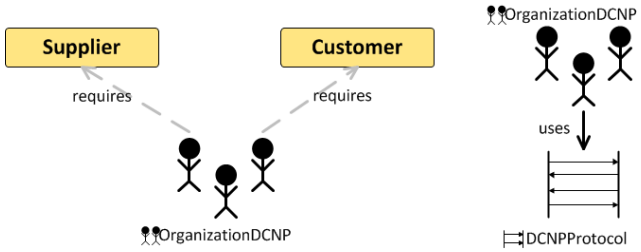


Figure 3: MAS (left) and organization (right) views (excerpts)

Besides the MAS view, an "organization view" is required to define which interactions can be performed within the organization i.e. one stage of the SC. In the collaborative DCNP case, these interactions are modeled by a protocol called *DCNPProtocol* – described in detail below – which contains all communication steps of the coordination mechanism (cf. Figure 1). Therefore, a *uses* relation between the *OrganizationDCNP* and the *DCNPProtocol* is modeled in the organization view in order to allow the application of this protocol to the different stages of the SC being coordinated (cf. Figure 3, right).

Following the inter-organizational communication defined in the collaborative DCNP concept, the *DCNPProtocol* can be divided into three major phases: communication of plans, conflict identification and the reaction upon conflict situations including negotiation processes for potential plan adaptions (cf. Figure 4). The protocol uses Agent Communication Language (ACL) messages in the communication process and prescribes the order of message exchange in the different interaction and negotiation phases. Within DCNP a set of negotiation states and message types is defined that were implemented by respective state transitions and message exchanges in the *DCNPProtocol*.

The inter-organizational coordination is started by the actor called *Initiator* who sends an *activation signal* to the *Participant* and thus initializes the whole coordination process. The initiator is a customer and hence initially an OEM in the coordinated SC. Since the concept covers multiple tiers of SCs, also BTO suppliers (as customers of BTS- or other upstream BTO-suppliers) may initiate the *DCNPProtocol* in order to coordinate their supplier network. The next protocol step is the exchange of the consumption plan i.e. the intended capacity utilization in the planning horizon. This results in a state transition of both actors (represented by so called sub-actors *UpdatedCustomer* and *UpdatedSupplier*). The exchanged plans are compared with the performance agreement in order to distinguish between regular and conflict situations. In case of no conflicts, the *DCNPProtocol* is terminated (indicated in Figure 4 by an arrow defining the state transition to the finalization phase).

In case of a performance agreement violation, the actual negotiation protocol for conflict resolution is started. The negotiation protocol requires several state transitions and local planning invocations, especially in order to account for the different stages of the conflict resolution process (e.g. adaption request sent, temporary reservation made and final reservation accepted). Due to the support of multi-tier networks, the *DCNPProtocol* is processed in a recursive manner. This allows BTO suppliers to coordinate their supply network before accepting definite changes to performance agreements on the customer side. After processing the negotiation protocol and coordinating the supply network (or in

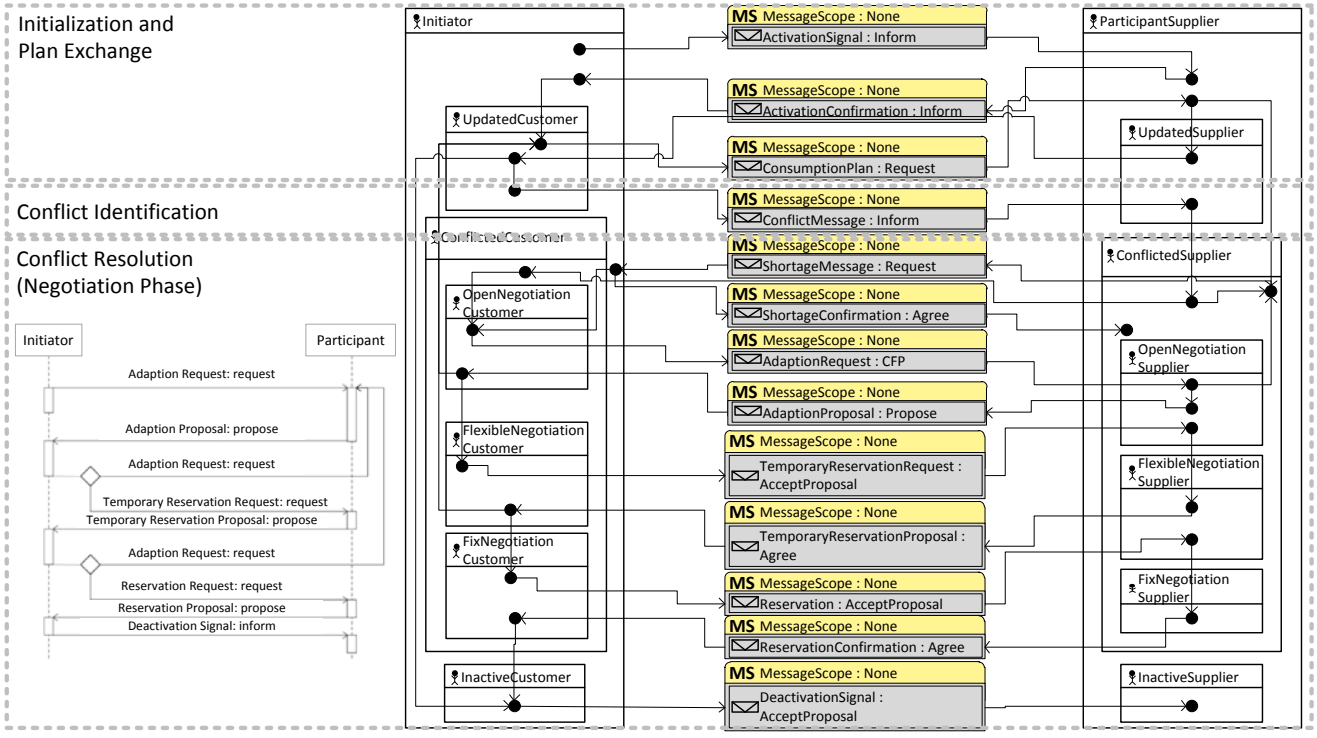


Figure 4: Phases in the collaborative DCNP concept and corresponding message exchanges in the protocol.

case of no conflicts), the *DCNPProtocol* terminates by sending a *DeactivationSignal*.

The collaborative DCNP concept defines several intra- and inter-organizational business processes that are performed for coordinating the SC. As mentioned above, these processes are modeled by defining plans that describe the behavior of agents in detail. These plans control the internal behavior and therefore connect local planning and decision processes to the inter-organizational communication processes as defined in the *DCNPProtocol*. Thus, besides the internal tasks to be performed at the different stages of the coordination process (plan exchange, monitoring and conflict resolution), the required procedures, with respect to production planning (primary demand determination, adjustment planning, local deficit calculation and proposal formulation) and procurement planning (secondary demand determination, shortage management and deficit calculation), were modeled in several plans in the agent view.

Modeling the intra- and inter-organizational processes of a CP concept by means of DSML4MAS constitutes a large part of the modeling process in the framework that is pursued by our research.[1] Since an MDD approach is followed, these models can be translated into source code allowing the execution of the coordination concept in a concrete SC scenario in the JADE runtime environment. The following chapter therefore describes further required steps in implementing and evaluating the collaborative DCNP concept.

4 Implementing the Collaborative DCNP Concept

The models of the collaborative DCNP in FRISCO define the basis for the implementation in an MAS runtime environment. The DSML4MAS development environment provides several translation rules that ultimately allow the generation of executable (agent) source code.

The models of the MAS as described in the previous chapter conform to the PIM4Agents metamodel (cf. [Hahn *et al.*, 2009]). The developers of the DSML4MAS environment defined mapping rules that allow a translation of corresponding platform independent MAS models to platform specific models and furthermore specified translations of the platform specific models to executable code. These two transformation steps therefore provide a mechanism to automatically generate JADE compliant agent code (in Java) based on the graphical models of the collaborative DCNP. In addition, the MDD approach allows for the inclusion of custom source code into the automatically generated code (e.g. in order to invoke local optimization) which is not changed by the transformation engine in case of model modifications.

[1] Besides the described models it is furthermore necessary to define the SC agents' knowledge in an ontology. This ontology is provided by the framework and already contains many SCM-related concepts to describe SCs, e.g. for sourcing relations, bill-of-material, resources etc. (cf. [Hellingrath *et al.*, 2009]). Extensions being necessary in this part of the framework in order to model a CP approach will not be described in detail here.

Besides the models of the collaborative DCNP described in the previous chapter, an implementation of this concept in one concrete SC scenario requires the structural definition of a SC i.e. the assignment of specific actors to their respective roles in the coordination mechanism. This scenario definition is specified in the "deployment view" of DSML4MAS. Figure 5 shows an example of a deployment diagram which is used to specify a two tier SC consisting of one OEM, one BTO supplier and two BTS suppliers.

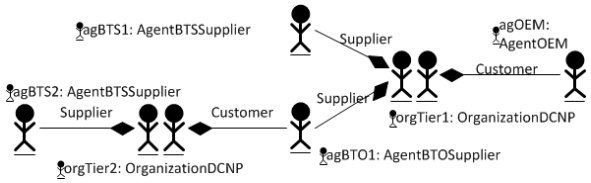


Figure 5: Scenario definition in the deployment view

The four agent instances are assigned with their respective roles to two organizations (*OrganizationDCNP*) which were specified in the organization view (see descriptions in chapter 3). These assignments define that the agents participate in the collaborative DCNP concept, i.e. the interactions in these organizations follow the *DCNPProtocol*. Since one *OrganizationDCNP* represents one stage of the SC being coordinated, two organizations are modeled in the deployment diagram in order to represent a multi-tier structure. The OEM/1st-tier relationship is therefore modeled by an organization called *orgTier1*; the 1st/2nd-tier relationship respectively by *orgTier2*. Based on this deployment diagram the agent source code is generated by the DSML4MAS transformation rules. The results represent the SC scenario and can be deployed on the JADE platform afterwards. The agents are capable of performing the collaborative DCNP concept in a runtime environment allowing its evaluation.

In order to show and evaluate the correct operation of the collaborative DCNP concept in the scenario, the MAS was extended by a mechanism to simulate the business processes and hence the DCNP protocol execution. This was achieved by modeling a "simulation protocol" (also in DSML4MAS) which all agents participate in and which is controlled by a "simulation agent". This allows triggering the agents each period, i.e. the intra- and inter-organizational processes of the collaborative DCNP are executed in the scenario consecutively over multiple periods.

Figure 6 shows a screenshot of the graphical visualizations of two agents which perform the collaborative DCNP concept in the exemplary SC: the agent depicted on the left side represents the BTS supplier on tier 2 (*agBTS2*). The agent interface shown on the right side in Figure 6 represents one of the two suppliers on tier 1 (*agBTO1*). Thus, *agBTO1* is supplied by *agBTS2* and in turn supplies the OEM (*agOEM*). One period was simulated, i.e. the ultimate customers requested products from the OEM, which afterwards communicated the resulting capacity demand to its suppliers (including *agBTO1*). Conflict situations were identified and the collaborative DCNP coordination mechanism started. Throughout the coordination processes, network-wide adaptations of the capacity corridors were

negotiated and agreed upon in order to meet the capacity demand as good as possible. The thick lines indicate deviations from the original performance agreements (thin lines) that result from the automated negotiation processes. The capacity demand from the *agOEM* required adaptations to the performance agreement between *agBTO1* and *agOEM* (right side in Figure 6). Changes to the capacity corridors of *agBTO1* furthermore affected its supply network, i.e. also *agBTS2*. This agent therefore had to increase the capacity corridors of its performance agreement with *agBTO1* in the planning horizon (left side in Figure 6).

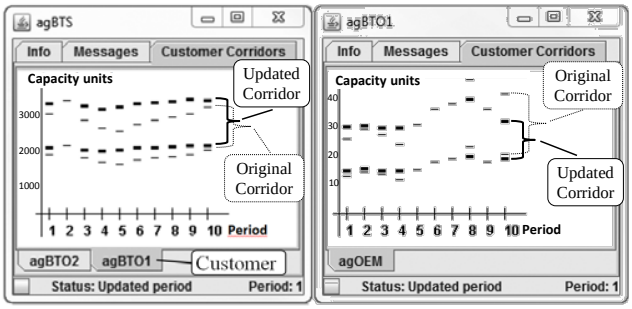


Figure 6: Annotated screenshot of agents in an exemplary scenario

In addition to the results of the coordination mechanism in form of agreed plan adaptations, the conducted inter-organizational communication processes were analyzed. By means of the JADE "sniffer agent", the messages exchanged were traced proving the correct operation of the *DCNPProtocol* over multiple tiers. These analyses therefore allow a first evaluation of the collaborative DCNP concept. Apparently, plan adaptations were performed in order to improve the match between demanded and offered capacity in the network. However, the evaluation component of FRISCO has to be further extended for a transparent and comprehensive monitoring of CP and SC performance indicators.

5 Conclusion

The research presented in this paper has two goals. First, the suitability of MAS concepts in the domain of CP and decentralized SC coordination is intended to be substantiated. This goal was achieved by modeling and implementing the collaborative DCNP coordination mechanism by means of concepts and methods from the MAS research domain. The structure of heterarchical SCs and the crucial inter-organizational processes in CP approaches proved to be reasonably representable by means of an MAS modeling environment. The different views provided by the used DSML4MAS allow an elegant modeling of all intra- and inter-organizational business processes of the collaborative DCNP concept. Based on these models, different SC scenarios can be defined to which the concept is easily applicable. The automated code generation from MAS models to executable Java code provides a major advantage in the consistent and re-usable implementation of CP concepts.

The second goal of the presented research addresses the usage of MAS concepts in a more general framework for modeling and evaluating arbitrary decentralized coordination mechanisms in heterarchical SCs. The implemented proof-of-concept strengthens the argumentation for this approach since requirements on the framework were shown to be satisfiable by an MAS-based concept. However, the framework is still under development and will be extended in order to allow both an efficient modeling and evaluation of arbitrary CP approaches. This especially requires research on the measurement of CP performance indicators and an extension of the framework's evaluation capabilities. In addition, the methodology that guides the CP modeling and evaluation processes will be elaborated further.

References

[Breiter *et al.*, 2009]. A. Breiter, T. Hegmanns, B. Hellingrath, and S. Spinler. Coordination in Supply Chain Management - Review and Identification of Directions for Future Research. In S. Voss, et al. *Logistik Management*, Physica-Verlag, Heidelberg, 2009.

[Hahn *et al.*, 2009]. C. Hahn, C. Madrigal-Mora, and K. Fischer. A platform-independent metamodel for multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 18(2): 239–266.

[Hellingrath *et al.*, 2009]. B. Hellingrath, C. Böhle, and J. van Hueth. A Framework for the Development of Multi-Agent Systems in Supply Chain Management. In R.H. Sprague. *Proceedings of the 42nd annual Hawaii International Conference on System Sciences*, pages 1530–1605, IEEE, 2009.

[Hellingrath and Hegmanns, 2010]. B. Hellingrath and T. Hegmanns. Dezentrales und kollaboratives Bedarfs- und Kapazitätsmanagement in build-to-order Produktions- und Logistiknetzwerken. In W. Delfmann. *Strukturwandel in der Logistik, Wissenschaft und Praxis im Dialog*, pages 11–31, DVV Media Group, 2010.

[Moyaux *et al.*, 2006]. T. Moyaux, B. Chaib-draa, and S. D'Amours. Supply chain management and multiagent systems: an overview. In B. Chaib-draa and J. Müller. *Multiagent-Based Supply Chain Management*, pages 1–27, Springer, Berlin, 2006.

[Nunes *et al.*, 2009]. I. Nunes, E. Cirilo, C.J.P. Lucena, J. Sudeikat, C. Hahn, and J.J. Gomez-Sanz. A Survey on the Implementation of Agent Oriented Specifications. In M.P. Gleizes and J.J. Gómez-Sanz. *Agent-oriented software engineering X*, pages 169–179, Springer, Berlin, 2009.

[Schuh, 2006]. G. Schuh. Produktionsplanung und -steuerung. Grundlagen, Gestaltung und Konzepte, Springer, Berlin, 2006.

[Stadtler, 2009]. H. Stadtler. A framework for collaborative planning and state-of-the-art. *OR Spectrum*, Vol. 31, No. 1, pages 5-30, Springer, 2009.

ARMO: Adaptive Road Map Optimization for Large Robot Teams

Alexander Kleiner*, Dali Sun* and Daniel Meyer-Delius* *

Abstract

Autonomous robot teams that simultaneously dispatch transportation tasks are playing more and more an important role in present logistic centers and manufacturing plants. In this paper we consider the problem of robot motion planning for large robot teams in the industrial domain. We present adaptive road map optimization (ARMO) that is capable of adapting the road map in real time whenever the environment has changed. Based on linear programming, ARMO computes an optimal road map according to current environmental constraints (including human whereabouts) and the current demand for transportation tasks from loading stations in the plant. We show experimentally that ARMO outperforms decoupled planning in terms of computation time and time needed for task completion.

1 INTRODUCTION

Recent trends in logistics and manufacturing clearly indicate an increasing demand for flexibility, modularity, and re-configurability of material flow systems. Whereas in the past plant installations have been used for decades without change, nowadays product life cycles and the demand for product variety rely on innovative technologies that allow to flexibly reconfigure automation processes without reducing their availability. Therefore, distributed and self-organized systems, such as teams of robots that autonomously organize transportation tasks, are playing an increasingly important role in present logistic centers and manufacturing plants.

Besides the task assignment problem, i.e., allocating robots to different tasks [14], another challenge in this domain is to efficiently coordinate the simultaneous navigation of large robot teams in confined and cluttered environments. In general, multiple robot motion planning can be solved by either considering the joint configuration space of the robots [2] or by deploying decoupled techniques that separate the problems of motion planning and coordination [10]. Whereas the first approach is intractable for large robot teams since the dimension of the joint configuration space grows linearly and thus the search space grows exponentially with increasing number of robots, the second approach yields typically sub-optimal solutions, for example, requiring the robots to perform larger detours in order to avoid collisions. Road map planners are a popular method for single robot planning in static environments [9] that compute during a pre-processing phase a connectivity graph in free configuration space that is then used for efficient path planning during runtime. However, dynamic domains, such as industrial environments, are more challenging due to permanent changes in the environment, e.g., due to the placement and removal of objects such as pallets and boxes, and the co-location of human workers.

In this paper we present adaptive road map optimization (ARMO) for large robot teams that is capable of adapting the road map in real time whenever the environment has changed. In short, the planner computes an optimal road map according to current environmental constraints (including human whereabouts) and the current demand for transportation tasks from the loading stations. We describe the environment of the robot with a spatial grid map in which a hidden Markov model (HMM) is used to represent dynamic changes [11]. From the continuously updated grid map the computation of a Voronoi Graph [4] is triggered whenever significant changes have been detected. The Voronoi graph, representing free space connectivity, is taken as a starting point to extract road segments (as shown in Figure 1) for the final road map. We use a Linear Programming (LP) approach for computing the optimal configuration of these segments with respect to minimal travel costs and maximal compactness of the network. Figure 1 depicts the re-arrangement of the road map after local changes of the environment have been detected. We show experimentally that ARMO outperforms decoupled planning in terms of computation time and time needed for task completion.

Kallman et al. used dynamic roadmaps for online motion planning based on Rapidly-exploring Random Trees (RRTs) [8]. Velagapudi et al. introduced a distributed version of prioritized planning for large teams where each robot plans simultaneously and re-plans in case a conflict has been detected [18]. Berg et al. presented a method for road map based motion planning in dynamic environments [16]. In contrast to our method, which learns changes of the environment online, their approach discriminates between static and dy-

* Department of Computer Science, University of Freiburg, Georges-Koehler-Allee 52, 79110 Freiburg, Germany, {kleiner,sun,meyerdel}@informatik.uni-freiburg.de

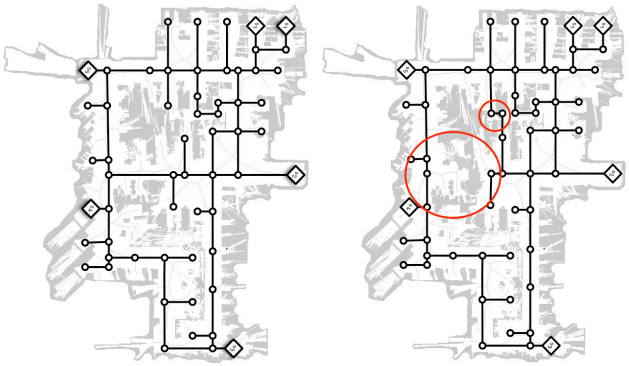


Figure 1: Motivating example: In industrial environments the map can locally change due to replaced objects, such as pallets, as well as gathering humans. Adaptive road map optimization facilitates the simultaneous navigation planning of large robot teams while respecting these changes.

namic objects, e.g. walls and robots, in advance, which might fail when also portions of the map have to be considered as dynamic. Bellingham et al. proposed a method for solving the cooperative path planning for a fleet of UAVs [3]. They formulate the task allocation problem as a mixed-integer linear program (MILP). Sud et al. developed an approach for path planning of multiple virtual agents in complex dynamic scenes [13]. They utilize first- and second-order Voronoi diagrams as a basis for computing individual agent paths. While computational efficient, their method does not focus on optimizing the global efficiency of the multi agent team.

The reminder of this paper is organized as follows. In Section 2 the problem is formally described and in Section 3 a description of the target system is provided. In Section 4 the algorithm for adaptively recomputing the road map are described, and in Section 5 results from experiments are presented. We finally conclude in Section 6.

2 PROBLEM FORMULATION

We consider the problem of coordinating the execution of delivery tasks by a team of autonomous robots, e.g., the transportation of crates containing goods, between a set of fixed stations $\mathcal{S}$. For each delivery task $d^{kl} \in \mathcal{D}(t)$ a robot has to be assigned to finalize the delivery by transporting the corresponding crate from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$. We assume that the assignment problem has been solved (e.g. as shown in our previous work [14]), and hence restrict our attention to the problem of solving the multiple robot motion planning problem as defined in the following. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be the set of n robots navigating simultaneously on a two-dimensional grid map. During planning, each robot has a start configuration $s_i \in \mathcal{C}_{free}$ and a goal configuration $g_i \in \mathcal{C}_{free}$, where $\mathcal{C}_{free}$ is the subset of configurations robots can take on without colliding with static obstacles. Note that in our case these configurations directly map to locations and orientations on the discrete grid map which are collision free given the footprint of the robot.

Figure 2: The target system: Robots equipped with convoyer and RFID reader for autonomously handling transportation tasks: (a) approaching a station for loading. (b) safe navigation among humans.

The problem is to compute for each robot $R_i \in \mathcal{R}$ a path $\pi_i : [0, T_i] \rightarrow \mathcal{C}_{free}$ such that $\pi_i(0) = s_i$ and $\pi_i(T_i) = g_i$ which is free of collisions with the trajectory π_j of any other robot $j \neq i$. Note that T_i denotes the individual path length of robot R_i.

We consider environments with dynamic obstacles such as pallets and larger crates that might change their locations over time. Therefore, $\mathcal{C}_{free}$ is a function of time which we denote by $\mathcal{C}_{free}(t)$. Note that we assume that $\mathcal{C}_{free}$ is static during each planning cycle.

3 SYSTEM OVERVIEW

Our system is based on the KARIS (Kleinskalige Autonomes Redundantes Intralogistiksystem) [7] platform developed by a joint effort of several companies and universities of the "Intralogistic Network" in south Germany. The long-term goal of this project is to deploy hundreds of these elements to solve tasks in intra-logistics and production, such as autonomously organizing the material flow between stations. The element has a size of 50×50 cm, a payload of 60 kg, and is capable to recharge its batteries via contact-less rechargers let into the ground. Furthermore, it contains a high precision mechanism for enabling automatic docking maneuvers, either with other elements or a loading station. Each element is equipped with a holonomic drive to facilitate docking behaviors and a conveyor for loading and unloading crates when docked with a loading station. The convoyer has an integrated RFID reader for directly reading from the crates their destination, e.g. the target station ID, when they are placed on the conveyer.

For the purpose of autonomous navigation the element is equipped with two SICK S300 laser range finders (LRFs) mounted in two opposing corners, wheel odometry, and an inertial measurement unit (IMU). Navigation is based on grid maps, which are generated from data collected by once steering a single robot manually through the environment. We use Monte-Carlo localization [6] with wheel odometry, IMU, and range readings from the two LRFs for localizing robots on the grid map. Furthermore, the typical hybrid architecture is deployed consisting of two components, which are a deliberative planning layer based on the grid map and a reactive safety layer based on LRF data directly. Figure 2 depicts the demonstration of the system during the *Logimat* fair in Stuttgart 2010. At the current stage, the system is capable of

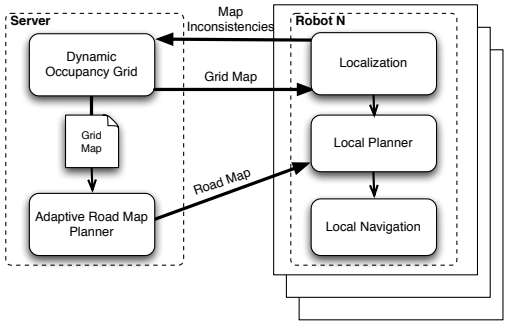


Figure 3: System Overview

safe autonomous navigation in human workspaces for team sizes of up to four robots.

The work presented in this paper has the goal to extend the planning system for the simultaneous navigation of large robot teams in dynamically changing environments. Figure 3 depicts the overall system architecture and modules of the considered extension. The localization module reports inconsistencies between sensor observations and the current grid map to the *Dynamic Occupancy Grid Module* which computes an updated version of the grid map [11]. The updated grid map is published to the localization module of each robot, and also to the *Adaptive Road Map Planner* (see Section 4) that computes a new road map, which is then published to the local planner of each robot. The local planner computes then based on the road map a path that is executed by the navigation module. The overview does not contain the mechanism for task allocation, i.e., to assign robots to delivery taks. In the current system this task is solved by the *contract net protocol* [12], however, also more sophisticated approaches, such as the one presented in our previous work [14] can be deployed.

4 ADAPTIVE ROADMAP PLANNER

In this section we describe the procedure for computing the adaptive road map given a dynamic occupancy grid map, a set of stations $s \in \mathcal{S}$, where $loc(s)$ denotes the location (x_s, y_s) of station s on the grid map, and a set of *delivery tasks* $\mathcal{D}$, where each $d^{kl} \in \mathcal{D}$ requires the routing of packages from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$.

4.1 Computation of the connectivity network

Our goal is to compute a road map that is optimal in terms of efficiency and compactness for the simultaneous routing of robots executing delivery tasks. For this purpose we first compute the Voronoi graph [4] from the dynamic grid map, which then serves as a basis for computing the connectivity network $\mathcal{C} = (V, E)$ consisting of nodes $v \in V$ that correspond either to station locations $loc(s)$ or crossings, and edges $e \in E$ that connect all stations and crossings on the map. The computation of $\mathcal{C}$ is carried out by three steps. First, we determine for each tuple $(i, j) \in \mathcal{S} \wedge i \neq j$ the set of alternative paths A_{ij} connecting station i and j on the Voronoi graph. Second, according to the method described in [5], we replace each A_{ij} by orthogonal straight lines (either horizontal or vertical) under the constraint that they have to be within a minimum safety distance to obstacles including the maximal extent of robots from their rotational center. Third, we add all straight lines to E while merging parallel lines if they exceed the double size of the robots. Besides station locations $loc(s)$, for each crossing line a node is created and added to V. Finally, we compute for each $e_{ij} \in E$ the maximal number of possible lanes w_{ij} for this connection according to the distance to the nearest obstacle, and the time needed to travel this segment c_{ij} according to its length.

4.2 Definition of the LP problem

Based on the connectivity network $\mathcal{C}$, we define our logistics problem similar to the *minimum cost flow problem* [1], however, with the difference that the number of lanes in both directions between two nodes and thus the capacities are variable. The goal is to find a network structure by which packages are optimally routed between the stations in the network. At each time there exists a set of simultaneous *delivery tasks* $d^{kl} \in \mathcal{D}(t)$ that require the routing of packages from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$. We denote by $b^{kl} = b(d^{kl})$ the requested throughput rate, i.e., the amount of packages per minute that have to be delivered from station k to station l.

Given the connectivity network C, we associate with each edge a cost c_{ij}, the maximal number of lanes w_{ij} allowed in the real world, and the capacity of a single lane connection u_{ij}. Whereas the cost c_{ij} expresses the time needed to travel from i to j, capacity u_{ij} expresses the maximal number of robots that can travel on this connection via a single lane at the same time without causing congestions. The number of lanes in both directions between two nodes i and j is expressed by the decision variables y_{ij} and y_{ji}, respectively. For example, $y_{ij} = 2, y_{ji} = 1$ denotes a single lane connection from node j to node i and a double lane from node i to node j. The quantity w_{ij} constraints the set of possible assignments to y_{ij} and y_{ji} according to the space available in the the real world. For example, if $w_{ij} = 4$, then some of the possible assignments are $(0, 0)$, $(0, 1)$, $(1, 0)$, $(2, 1)$, $(1, 2)$, $(2, 2)$, ... In general, it has to be assured that $y_{ij} + y_{ji} \leq w_{ij}$. Note that there exists the same limit in both directions and thus $w_{ij} = w_{ji}$.

The decision variables x_{ij}^{kl} define the flow assigned to an edge due to the delivery from k to l. The total flow x_{ij} has to be bigger or equal to zero and below the maximal flow $u_{ij}y_{ij}$, where u_{ij} is the capacity of a single lane and y_{ij} the number of activated lanes.

We associate for each delivery task d^{kl} the requested throughput $b(i)$ with the respective station nodes i. For each node $i \in V$, $b(i) = b^{kl}$ if $i = k$, i.e., vertex i is a source, and $b(i) = -b^{kl}$ if $i = l$, i.e., vertex i is a sink. All other nodes for which $b(i) = 0$ are functioning as transition nodes. The problem formulation can then be stated as follows:

$$\text{Minimize} \sum_{(i,j)} \sum_{k} \sum_{l} c_{ij} x_{ij}^{kl} + \sum_{(i,j)} u_{ij} y_{ij} \tag{1}$$

subject to:

$$\sum_{j:(j,i)} x_{ij}^{kl} - \sum_{j:(i,j)} x_{ji}^{kl} = \begin{cases} -b^{kl}(i) & (i = k) \quad \forall\, i,k,l \\ b^{kl}(i) & (i = l) \quad \forall\, i,k,l \\ 0 & otherwise. \end{cases} \tag{2}$$

$$y_{ij} + y_{ji} \leq w_{ij} \quad \forall\, (i,j), \tag{3}$$

$$x_{ij}^{kl} \geq 0 \quad \forall\, (i,j), k, l, \tag{4}$$

$$\sum_{k} \sum_{l} x_{ij}^{kl} \leq u_{ij} y_{ji} \quad \forall\, (i,j), k, l \tag{5}$$

$$\sum_{j:(i,j)} x_{ji}^{kl} \leq C_{max} \quad (i \neq k \wedge i \neq l) \quad \forall\, i,k,l \tag{6}$$

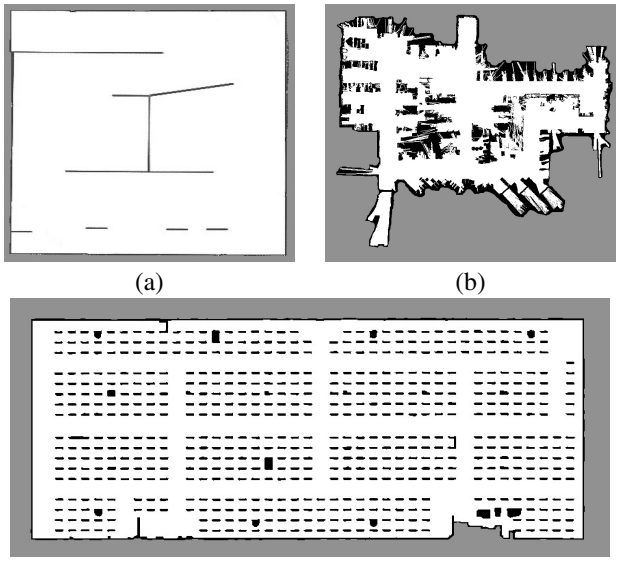
Equation 1 minimizes over the total travel costs and the physical space occupied by the road network. Equation 2 enforces the flow conservation in the network, i.e., the summed flow from all incoming edges $j : (j,i)$ and all outgoing edges $j : (i,j)$ has to be equal $-b_{kl}$ if i is a sink, b_{kl} if i is a source, and zero otherwise. Equation 3, Equation 4, and Equation 5 are constraining the maximal number of lanes, minimal and maximal flow, respectively. Finally, Equation 6 ensures that the total flow through crossings does not exceed the maximal crossing capacity C_{max} which depends on the spacial size of crossings, i.e., how many robots can be located there at the same time. Note that delivery tasks for which the node operates as source or sink have no influence on the capacity.

The above formulation can efficiently be solved by linear programming solvers, such as CPLEX, when defining the decision variables x_{ij}, y_{ij} by continuous values and rounding up the y_{ij} from which then the road map can directly be constructed. Furthermore, we yield for each delivery task d^{kl} a subset of edges from the road map having positive flow assignments $x_{ij}^{kl} > 0$. These quantities are directly utilized by the local planner (see Section 3) for extracting individual robot plans by finding the shortest path on the road map by the following successor state expansion: For each node i, we perform random sampling over all outgoing edges weighted according to their normalized flow values x_{ij}^{kl}. If there exists only one edge with $x_{ij}^{kl} > 0$ for node i, the edge is expanded directly. Finally, the local navigation module follows this plan while coordinating locally at crossings with other robots when needed.

5 EXPERIMENTAL RESULTS

The system has been tested in several different environments. Figure 4 depicts some of these environments that were used for the results presented in this paper. The *PLANT* map has a size of $51m \times 56m$, the *ASE* map a size of $94m \times 82m$, and the KNO map a size of $88m \times 43m$. On each map we defined locations of loading stations: 8 on PLANT, 16 on ASE, and 8 on KNO.

The robot platform shown in Figure 2 has been presented during the *Logimat* fair in Stuttgart, 2010, where the task of



(a) (b) (c)

Figure 4: Grid maps utilized for experiments: (a) the *PLANT* map generated from a simulated environment, (b) the *ASE* map generated in a real logistic center, (c) the *KNO* map generated in a large distribution center.

the robot team was to deliver freshly prepared coffee cups to visitors waiting at the delivery stations, and to return used cups back to the coffee kitchen. During this demonstration up to four robots were continuously running for three days without any interruption. The robots were driving in average four kilometers per day without causing collisions or deadlocks. Due to the small team size we utilized for this demonstration a decoupled planning technique together with the local navigation module. In the following a comparison with large robot teams between ARMO and the decoupled technique based on prioritized planning from Berg and colleagues [15] will be presented. In prioritized planning, robot trajectories are planned iteratively after a pre-defined priority scheme. When planning for the i'th robot trajectories of the i-1 robots that were planned previously are considered as dynamic obstacles. Berg and colleagues define the *query distance* as the distance for each robot to reach its goal configuration on the shortest path when ignoring the other robots. In order to minimize the maximum of arrival times, priorities are assigned according to this distance: the longer the query distance the higher the priority assigned to a robot. The planner is complete under the assumption that start and goal locations of each robot are so called *garage configurations*, i.e., configurations that are not part of $\mathcal{C}_{free}$ of any other robot. The method efficiently avoids the intractable computation of $n!$ possible priority schemes, however, requires at least $|\mathcal{R}|$ sequential calls of the motion planner.

We utilized the Stage software library [17] for simulating large robot teams. In our experiments we used the same navigation software that is used on the real robots together with a model of our real platform, including the simulation of laser beams and odometry. One advantage of Stage is that it allows

Map	#Rob.	Method	# C	(m/s)	CTime (s)
ASE	20	ARMO	1432	0.47	969
		PRIO	2142	0.43	926
	50	ARMO	5040	0.37	545
		PRIO	10625	0.28	631
	100	ARMO	8307	0.3	369
		PRIO	16983	0.2	496
PLANT	20	ARMO	1471	0.42	628
		PRIO	1346	0.38	610
	50	ARMO	5563	0.34	426
		PRIO	11601	0.25	481
	100	ARMO	15145	0.22	383
		PRIO	107700	0.21	874
KNO	20	ARMO	506	0.38	1383
		PRIO	5638	0.35	1346
	50	ARMO	2951	0.31	815
		PRIO	70799	0.16	1371
	100	ARMO	7729	0.29	513
		PRIO	102836	0.11	1167

Table 1: Comparing prioritized planning (PRIO) with adaptive road map optimization (ARMO).

to build simulation worlds directly from grid maps that were generated from real environments. For the following experiment we used the grid maps shown in Figure 4.

We generated 100 delivery tasks for each map that were handled by 20, 50, and 100 robots during different runs. Table 1 provides the results from comparing prioritized planning (PRIO) with adaptive road map optimization (ARMO) on different maps with different numbers of robots. We measured the number of conflicts (C) of the optimal path in $\mathcal{C}_{free}$ with trajectories of the other robots. In the case of ARMO these were the situations in which a robot had to wait for other robots before entering a segment, and in the case of prioritized planning these were the situations were robots had to plan around a conflicting path of a higher prioritized robot. Furthermore, we measured the average velocity of all robots (avg. v) and the total time needed by all robots to complete the task (CTime). As can be seen from Table 1 and Figure 5, while leading to slightly longer completion times for small robot teams, ARMO notably reduces this time when the team size increases. This is also reflected by the number of conflicts and the average velocities of the robots. Prioritized planning minimizes the final completion time after a heuristically determined order, whereas LP-based planning in ARMO minimizes the global flow of robots, leading to a more efficient distribution of the vehicles over time.

The computation times of both methods were measured in seconds on an Intel DualCore running at 2.13 GHz. We measured for prioritized planning with 50 robots an average computation time of 0.03 ± 0.03 on PLANT, 0.05 ± 0.04 on ASE, and 0.1 ± 0.17 on KNO, and with 100 robots 0.1 ± 0.08 on PLANT, 0.13 ± 0.08 on ASE, and 1.1 ± 0.7 on KNO. ARMO required for the road map computation $0.9 + 0.6$ on PLANT, $0.82 + 0.84$ on ASE, and $1.2 + 10.3$ on KNO, where the first number denotes the time for extracting the fully connected graph, and the second number the time for solving the

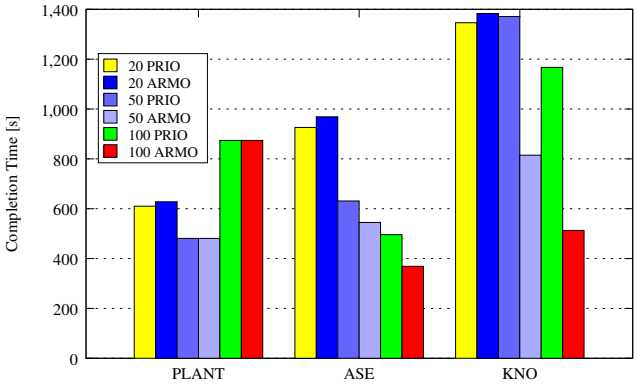


Figure 5: Comparing the CTime of prioritized planning (PRIO) and adaptive road map optimization (ARMO).

LP problem. Within each planning cycle ARMO needed in average only 0.002 on PLANT, 0.004 on ASE, and 0.01 on KNO for any number of robots. In summary, the number of robots has nearly no effect on the computation time needed by ARMO, however, we measured a significant growth of the time needed by prioritized planning. On the contrary, ARMO requires much more time for computing the road map when the environment is very large and complex, such as the KNO map, which however needs only to be performed at low frequency, i.e., when the environment was significantly changed.

We also evaluated ARMO with respect to dynamic changes of the grid map. For this purpose we modified the ASE map step wise by adding successively obstacles that were updated in the map by the dynamic occupancy grid approach. Figure 6 depicts two snapshots taken at successive points in time. As can be seen, the road map adjusts to the changes at the cost of higher completion times. For 100 robots the completion time increased from 378s (no modifications) to 410s (first modification) and 420s (second modification). We performed several more experiments for evaluating the adaptivity of our approach. Also after changing the distribution of delivery tasks between the stations, the road map dynamically adjusted by removing or adding links between the stations. Note that in this case only the LP solver is restarted without re-computing the connectivity network $\mathcal{C}$.

6 CONCLUSION

We proposed an adaptive road map planner based on a linear programming formulation which can be used for motion planning of large robot teams in dynamically changing environments. Experimental results have shown that ARMO leads to more efficient multi-robot plans than decoupled techniques while keeping the demand for computational resources low. In fact, the computation time needed by ARMO depends mainly on the complexity of the environment rather than on the number of robots. We believe that the computation of the road map could further be improved by splitting the map into independent areas that are interconnected via *fixed* crossing points similar to the stations. Then, only a part of the road map would have to be recomputed after local changes have been detected.

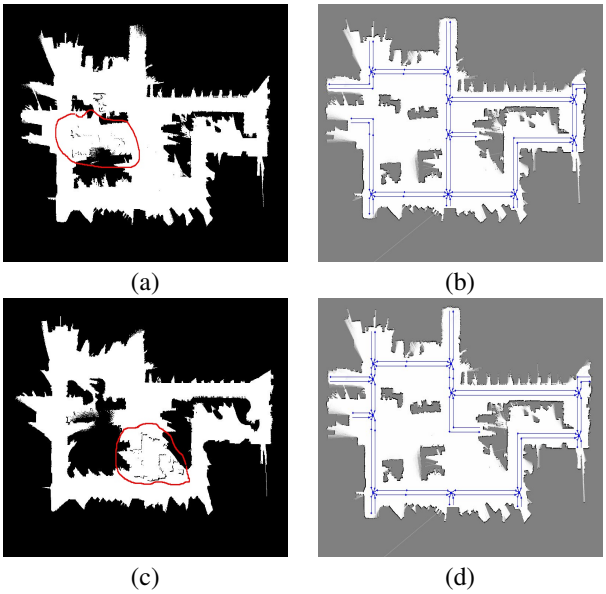


Figure 6: Adjustment of the road map according to dynamic changes in the map (a,c) source of disturbance and (b,d) resulting modifications reflected in the road map.

Furthermore, we have shown that ARMO is adaptive to dynamic changes in the map, i.e., the road map is reconstructed accordingly, whereas changes in the map are detected by dynamic grid maps, an extension of conventional grid maps.

We conducted several more experiments and conclude that our method is capable to efficiently solve a wide variety of problems. One restriction of our current implementation is the fact that our road map planner only returns a solution when the overall throughput demanded by the stations can be routed given the environmental constraints, i.e., does not exceed the capacity of the network. One future extension will be to introduce priorities for deliveries and to construct the network from a subset of tasks sampled according to their priority in case the requested throughput is higher than the capacity of the network. Furthermore, when a larger number of real robots is available, ARMO will be used with the real platform deployed in one of the logistic centers of our partners.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*, volume 1. Englewood Cliffs, N.J.: Prentice Hall, 1993.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International journal of robotics research*, 10:628–649, 1991.

[3] J.S. Bellingham, M. Tillerson, M. Alighanbari, and J.P. How. Cooperative path planning for multiple uavs in dynamic and uncertain environments. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 3, pages 2816 – 2822 vol.3, 2002.

[4] H. Choset, , and Burdick J. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2), 2000.

[5] Xavier Décoret and François X. Sillion. Street Generation for City Modelling. In *Architectural and Urban Ambient Environment*, Nantes France, 2002.

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[7] H. Hippenmeyer, K. Furmans, T. Stoll, and F. Schönung. Ein neuartiges Element für zukünftige Materialflusssysteme. *Hebezeuge Fördermittel: Fachzeitschrift für Technische Logistik*, (6), 2009.

[8] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 5, pages 4399–4404, 2004.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566 –580, August 1996.

[10] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[11] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard. Temporary maps for robust localization in semi-static environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, Taipei, Taiwan, 2010.

[12] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

[13] A. Sud, E. Andersen, S. Curtis, M.C. Lin, and D. Manocha. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):526 –538, 2008.

[14] D. Sun, A. Kleiner, and C. Schindelhauer. Decentralized hash tables for mobile robot teams solving intra-logistics tasks. In *Proc. of the 9th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 923–930, Toronto, Canada, 2010.

[15] J.P. van den Berg and M.H. Overmars. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, pages 430–435, 2005.

[16] J.P. van den Berg and M.H. Overmars. Roadmap-based motion planning in dynamic environments. *Robotics, IEEE Transactions on*, 21(5):885–897, 2005.

[17] R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2):189–208, 2008.

[18] P. Velagapudi, K. Sycara, and P. Scerri. Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4603–4609. IEEE.

HCOP : Modeling Distributed Constraint Optimization Problems with Holonic Agents

Fernando J. M. Marcellino, Jaime S. Sichman
Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP) - Universidade de São Paulo (USP)
Av. Luciano Gualberto, 158 – trav. 3
05508-970 – São Paulo – SP – Brasil
fmarcellino@usp.br, jaime.sichman@poli.usp.br

Abstract

This paper defines a new distributed optimization problem, called Holonic Constraint Optimization Problem (HCOP). It is based on the concepts of Distributed Constraint Optimization Problem (DCOP) and Holonic Agents. We present the background theory and the formalization of a HCOP, which rather than a mere generalization of a DCOP, represents a distinct paradigm. We also propose a meta-algorithm, called HCOMA, to solve this kind of problem, where several available DCOP algorithms, or even centralized algorithms, can be embedded and integrated in such a way to obtain the most fitting configuration for each case. In addition, a motivating application in the oil supply chain domain is presented in order to illustrate the new approach.

1 Introduction

Constraint satisfaction and optimization are powerful paradigms that model a large range of tasks like scheduling, planning, optimal process control, etc. Traditionally, such problems were gathered into a single place, and a centralized algorithm was applied in order to find a solution. However, problems are sometimes naturally distributed, so Distributed Constraint Satisfaction (DisCSP) was formalized by Yokoo et al. in [Yokoo *et al.*, 1992]. Here, the problem is divided among a set of agents, which have to communicate with each other to solve it. More recently this paradigm was extended to constraint optimization by replacing the logical constraints with valued ones, and it was formalized as a Distributed Constraint Optimization Problem (DCOP) [Modi *et al.*, 2006]. In general, an optimization problem is much harder to solve than a DisCSP, as the goal is not just to find any solution, but the best one.

If we analyze some real constraint optimization problems, we can notice that they own a *recursive* nature, which is not currently exploited by the available optimization frameworks and their associated algorithms. An example of this kind of problem is the supply chain management. Usually each entity in the chain is likely to act in its best interests to optimize its own profit. However, in general, that doesn't meet the goal of the optimization of the entire supply chain. On the other hand, the complexity of the whole chain integration makes the development of a single centralized system an unfeasible task. In addition, even if it were possible, the frequent and unforeseeable changes in the business environment would make the results of such a system obsolete and useless very fast.

This paper defines a Holonic Constraint Optimization Problem (HCOP) as a new paradigm to model distributed optimization problems which meet those features. Its main motivation is modeling such problems through the integration of solvable subproblems into which they may be naturally partitioned. Sections 2 and 3 synthesizes the basic concepts involved in the work, whereas section 4 introduces a problem of the oil supply chain industry, which was the original motivation for the proposed model. Section 5 describes and formalizes the HCOP, and suggests a meta-algorithm, called HCOMA, for its solution. Section 6 characterizes the problem presented in section 4 as a HCOP, and shows the advantage of this approach. Finally, the paper concludes with a summary of the results, and an outlook on future research activities.

2 Holonic Agents

MutiAgent System (MAS) has become a natural tool for modeling and simulating complex systems. However, in those systems there usually exist a great number of entities interacting among themselves, and acting at different *abstraction levels*. In this context, it seems unlikely that MAS will be able to faithfully represent complex systems without multiple granularities. That's why holonic systems have attracted the attention of researchers [Hilaire *et al.*, 2008]. The term holon was coined by Arthur Koestler [Koestler, 1967], based on the Greek words *holos* for whole and *on* for part. Thus, a holon is a self-similar or fractal structure that consists of several holons as components, and is itself a part of a greater whole. A holon (superholon) is composed of other holons (members or subholons) and should meet three conditions: (i) to be stable, (ii) to be autonomous and (iii) to be able to cooperate. Thus, according to Koestler [Koestler, 1967] a *holarchy* is a hierarchy of self-regulating holons that function first as autonomous wholes in supra-ordination to their parts, secondly as dependent parts in sub-ordination to controls on higher levels, called *echelon*s, and thirdly in coordination with their local environment.

Gerber et al. [Gerber *et al.*, 1999] propose three types

of structures for holons, which vary with respect to the autonomy of the members. The moderated group is the intermediary structure, which was chosen for this work due to its greater flexibility. According to [Hilaire *et al.*, 2008] it specifies a holonic organization with three main roles: *head* role players are moderators of the holon, whereas represented members have two possible roles: *part*, whose players belong to only one superholon, and *multipart*, where subholons belong to more than one superholon. The *head* represents the shared intentions of the holon and negotiates them with agents outside the holon. The remainder of the holon, i.e. the set of parts and multiparts, is called *body.*

3 Distributed Constraint Optimization Problem (DCOP)

DCOP is a formalism that can model optimization problems distributed due to their nature. These are problems where agents try to find assignments to a set of variables that are subject to constraints. It is assumed that agents optimize their cumulated satisfaction by the chosen solution. This is different from other related formalisms involving self-interested agents, which try to maximize their own utility individually. Thus, the agents can optimize a global function in a distributed fashion communicating only with neighboring agents, and even in a asynchronous way.

3.1 Formalization

According to [Petcu *et al.*, 2007], a DCOP can be defined as a tuple $(A,\ V,\ D,\ F)$ where :

- $A = \{a_1, \ldots, a_n\}$ is a set of n agents,
- $V = \{v_1, \ldots, v_n\}$ is a set of n variables, one per agent,
- $D = \{D_1, \ldots, D_n\}$ is a set of finite and discrete domains each one associated with the corresponding variable,
- $F = \{f_1, \ldots, f_m\}$ is a set of valued constraintsf_i, where $f_i : D_{\alpha_1} \times \ldots \times D_{\alpha_k} \rightarrow \mathbb{R},\ \alpha_k \in \{1 \ldots n\}$

The goal is to find a complete instantiation V^* for all the variables $v_i$that maximizes the objective function defined as

$$F = \sum_i f_i$$

3.2 Available Algorithms

The main complete algorithms developed for DCOP are:

ADOPT It is a backtracking based bound propagation mechanism [Modi *et al.*, 2006], which operates completely decentralized, and asynchronously. Its drawback is that it may require a very large number of small messages, thus producing considerable communication overhead.

OptAPO It is a hybrid between decentralized and centralized methods [Mailler and Lesser, 2004]. It operates as a cooperative mediation process, where agents designated as mediators centralize parts of the problem in dynamic and asynchronous mediation sessions. Message complexity is significantly smaller than ADOPT's. However, it may be inefficient with some mediators solving overlapping problems. Furthermore, the dynamic nature of the mediation sessions make it impossible to predict which part of the problem will be centralized.

DPOP It is an algorithm based on dynamic programming [Petcu and Faltings, 2005] as an evolution of the DTREE algorithm [Petcu and Faltings, 2004] for arbitrary topologies even with cyclic graphs. It generates only a linear number of messages, which, however, may be large and require large amounts of memory, up to space exponential. Therefore it was extended later, and a new hybrid algorithm called PC-DPOP was developed [Petcu *et al.*, 2007], that uses a customizable message size and amount of memory.

4 Motivating Scenario

As discussed in the introduction, an example of actual distributed optimization problem with a recursive organization is the supply chain management. Let us consider an oil company, which may be a single verticalized petroleum enterprise or a set of cooperating companies of the oil business. That enterprise system can purchase from the spot market (SM), which satisfies any extra demands of crude oil and its derivatives at higher prices. In the same way, SM can buy any exceeding inventories of those items at lower prices. A holonic model can be built according to geographical criteria, taking into account the transport integration. Thus there is a global holon, which comprises several continent holons, which are in their turn made up of region holons. These last holons may contain subholons like refineries, which are responsible for the production of oil derivative products, distribution terminals, which store those products, and oil extraction platforms, that yield crude oil (raw material). All the areas are connected by transportation modals, like ships and pipelines, and each area owns a specific logistic entity, which is responsible for planning the transportation of products.

In general, the refineries own their specific centralized optimization system for production planning, whereas the logistics of each echelon also has its respective optimization system for the corresponding transport planning. However, the different systems are not conveniently integrated to allow a global optimization. Figure 1 depicts the holonic echelons of this problem [Marcellino and Sichman, 2010].

5 Holonic Constraint Optimization Problem (HCOP)

Some distributed constraint optimization problems have a hierarchical and recursive structure, which is called *holarchy*. That organization is characterized by entities with great cohesion with respect to their fellows, but only a coupling relationship with their parents and childs along the hierarchy. Therefore, that kind of modeling allows that the optmization problem may be partitioned into a set of smaller optimization problems, which, although not independent from each other, present such a low coupling level that enable some parallelism. In addition it makes it easier to tackle the complexity of the whole system, which is modeled through a simpler model that repeats itself recursively throughout the complete model.

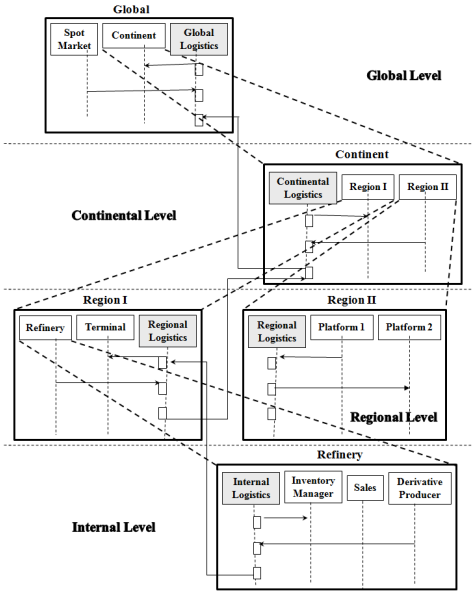


Figure 1: Diagram of an oil supply chain holonic model

5.1 Description

The HCOP consists of a set of agents that are called holons. Those holons are distributed into different abstract levels which are named *echelons*. By definition, a holon contains other holons (its subholons) and is part of another holon (its superholon). However, the most fundamental echelon ($\eta = 0$) comprises only atomic holons, i.e., a conventional agent that doesn't contain any other. Each holon is responsible for a variable. In the case of the atomic holon, it is a *decision variable*, which is an independent variable in the same sense of a DCOP variable. On the other hand, each holon belonging to a higher echelon ($\eta > 0$) is associated with an *emergent variable*, which is dependent on the internal variables of that holon, i.e., the variables associated with its subholons. Such dependency is specified by an *emergent function*.

The holonic organization adopted in this work classifies the subholons into 2 roles : *head*, which is unique for each holon, and *part*, which may be one or more. Here we don't consider the *multipart* role, as it will be seen later. It is also assumed that the *head* holons are atomic in all the echelons, for the sake of the model elegance, avoiding a recursive overload of that kind of holon as η increases. Due to the distinctive behavior of the *head* holon, which is responsible for the communication with the outside world, it is natural to set it apart from the remainder of the holon, which comprises all the *part* holons and is called *body*.

The internal strong cohesion of the holons, and the less intense coupling between a holon and its superholon or subholon, make it possible to view a HCOP as a partition of coupled optimization problems (OPs). Basically each holon may map to a corresponding OP.

5.2 Formalization

The HCOP is formalized as a tuple (H, R, V, D, E, F) where :

- $H = \{H_0, \ldots, H_{\eta_{max}}\}$, where η_{max} is the highest echelon, $H_\eta = \{h_{\eta 1}, \ldots, h_{\eta N_\eta}\}$ is the set of holons of the echelon η, H_0 is the set of atomic agents h_{0i} of the fundamental echelon, and $H_{\eta_{max}} = \{h_{\eta_{max} 1}\}$ contains a single holon (*global* holon);
- $R = \{r_1, \ldots, r_R\}$ is the set of relations between the holons, where $r_i$is one of the two primal relations :
  - $headOf_\eta : H'_\eta \rightarrow H_{\eta+1}$
  - $partOf_\eta : H'_\eta \rightarrow H_{\eta+1}$
    where $\eta \in \mathbb{N}, \eta < \eta_{max}, H'_\eta \subset H_\eta$

  Other important relations derived from these are :
  - $subholonOf_\eta : H_\eta \rightarrow H_{\eta+1}, \eta < \eta_{max}$, where $subholonOf_\eta \equiv headOf_\eta \cup partOf_\eta$
  - $superholonOf_\eta : H_\eta \rightarrow H_{\eta-1}, \eta > 0$, where $superholonOf_\eta \equiv subholonOf_\eta^{-1}$
- $V = \{V_0, \ldots, V_{\eta_{max}}\}$, where $V_\eta = \{v_{\eta 1}, \ldots, v_{\eta N_\eta}\}$ is the set of variables of echelon η (a variable per holon);
- $D = \{D_0, \ldots, D_{\eta_{max}}\}$, where $D_\eta = \{D_{\eta 1}, \ldots, D_{\eta N_\eta}\}$ is the set of discrete and finite domains associated with each variable of echelon η;
- $E = \{E_1, \ldots, E_{\eta_{max}}\}$, $E_\eta = \{E_{\eta 1}, \ldots, E_{\eta N_\eta}\}$ is the set of emergence functions of echelon η (one per holon, but the atomic), $E_{\eta i} : D_{\eta-1\alpha_1} \times \ldots \times D_{\eta-1\alpha_{B_{\eta i}}} \rightarrow D_{\eta i}$, where $B_{\eta i}$ is the *body* size of the holon $h_{\eta i}$, so that $v_{\eta i} = E_{\eta i}\left(v_{\eta-1\alpha_1}, \ldots, v_{\eta-1\alpha_{B_{\eta i}}}\right)$, where the domain is the cartesian product of the internal variables of holon $h_{\eta i}$, and $v_{\eta i}$ its emergent variable;
- $F = \{F_0, \ldots, F_{\eta_{max}}\}$, and $F_\eta = \{F_{\eta 1}, \ldots, F_{\eta N_\eta}\}$, $F_{\eta i} = \{f_{\eta i 1}, \ldots, f_{\eta i M_{\eta i}}\}$ is the set of $M_{\eta i}$ valued constraints between the members of holon $h_{\eta i}$, and $f_{\eta i j} : D_{\eta-1\alpha_1} \times \ldots \times D_{\eta-1\alpha_{M_{\eta i j}}} \rightarrow \mathbb{R}$, whose domain is the cartesian product of the $M_{\eta i j}$ variables$\{v_{\eta-1\alpha_1}, \ldots, v_{\eta-1\alpha_{M_{\eta i j}}}\}$, which is a subset of the set of internal variables of holon $h_{\eta i}$.

The goal is to find a complete instantiation V^* for all variables $v_{\eta i}$ that maximizes the objective function defined as

$$\mathcal{F} = \sum_{\eta=0}^{\eta_{max}} \sum_{i=1}^{N_\eta} \sum_{j=1}^{M_{\eta i}} f_{\eta i j} \qquad (1)$$

That definition reflects the holonic feature that there is no direct constraint f between two *part* subholons belonging to different superholons.

Since each holon $h_{\eta i}$ is responsible for an emergent variable $v_{\eta i}$ via its *head* agent, if it is taken into account the emergence function $E_{\eta i}$, it is possible to say that the agent *head* is connected by an n-ary constraint $c_{\eta i}$ with all the members of its holon, so that the following equation must be true :

$$v_{\eta i} = E_{\eta i}\left(v_{\eta-1\alpha_1}, \ldots, v_{\eta-1\alpha_{B_{\eta i}}}\right) \qquad (2)$$

5.3 Holonic Constraint Optimization Meta-Algorithm (HCOMA)

As already said, a HCOP can be seen as a holarchy whose holons may map to corresponding OPs. Each of these problems may be represented by a DCOP, or a centralized OP in the case of a holon with greater internal cohesion. Thus HCOP is a distributed OP, which may be modeled as a hybrid network of distributed and centralized optimization subproblems. Therefore, to take advantage of that feature, it is more appropriate a meta-algorithm for a HCOP, rather than a single algorithm. Thus, it is possible to embed into the more abstract framework different DCOP algorithms, or centralized optimization algorithms, in such a way to obtain the most fitting possibility for each case.

Since the holonic organization which was considered in this work does not include *multipart* holons, the macro graph made up of the several holons has a tree structure. In fact, its a connected graph without cycles. Therefore, it was developed a meta-algorithm, which was based on the DTREE algorithm [Petcu and Faltings, 2004]. Such a choice was due to the nature of that algorithm, which is free of backtracking, and hence evolves ininterruptedly upwards and then downwards, in a way compliant with the necessary independence between the optimization subproblems of the HCOP. On the other hand, algorithms like Adopt [Modi *et al.*, 2006] present a behavior which would interweave the holarchy echelons during the solving process and make that decoupling very hard.

At a first glance the exclusion of *multipart* holons seems an oversimplification, which aims at the reduction to a tree structure. However, in the same way DTREE evolved to DPOP (vide subsection 3.2) by arranging the relevant graph as a pseudotree, what is possible for any graph, it is straightforward to adapt HCOMA accordingly. Thus that enhancement would include *multipart* holons and support a general topology, keeping the backtracking free trait.

The proposed meta-algorithm has 3 phases, which are described in Algorithm 1. It is assumed that a generic and trustworthy optimization algorithm, distributed or centralized, is available in the scope of each pertinent holon. However, it must respect the protocol specified in Algorithm 2. For the sake of simplicity and readability, it was used another derived relation, as well as the predicate *head*, which are defined as :

$$headOfPart_\eta \equiv headOf_\eta(superholonOf_\eta) : H'_\eta \rightarrow H''_\eta$$

$$head(h_{\eta i}) := \exists h_{\eta+1\,k} \in H_{\eta+1} | \, h_{\eta i} = headOf(h_{\eta+1\,k}),$$
where $\eta \in \mathbb{N}, \eta < \eta_{max}, H'_\eta \subset H_\eta, H_\eta = H'_\eta \cup H''_\eta$

The phase 1 is a bottom-up process, which starts from the atomic holons and propagates upwards up to the global holon. In phase 2 the global holon owns the maximum utilities associated with each value of its emerging variable. That means it has the maximum values of the global objective function for each value of its variable. Then it will choose the highest value, which will represent the optimum value of the global objective function, whereas the associated value of its variable is the first assignment of the solution. Finally, in phase 3 the global holon will send the index of that solution, regard-

Algorithm 1 HCOMA - Holonic Constraint Optimization Meta-Algorithm

1: HCOMA(H, R, V, D, E, F)
2: **Phase 1: Utility Computation**
3: **for all** $h_{0i} \in H_0$ **and not** $head(h_{0i})$ **do**
4: **for all** $ind \in \{1, \ldots, \mid D_{0i} \mid\}$ **do**
5: $BestUtil_{0i}[ind] \leftarrow 0$
6: **end for**
7: send READY_msg($BestUtil_{0i}$, i) to $headOfPart_0(h_{0i})$
8: **end for**
9: **return**
10:
11: READY_msg_handler($BestUtil$, j){by holon $h_{\eta-1\,k}$}
12: $h_{\eta\,i} \leftarrow superholonOf_{\eta-1}(h_{\eta-1k})$
13: **for all** $ind \in \{1, \ldots, \mid D_{\eta-1j} \mid\}$ **do**
14: $BestUtilBody_{\eta\,i}[j][ind] \leftarrow BestUtil[ind]$
15: **end for**
16: **if** READY_msg received from $h_{\eta-1\,l}, \forall l, h_{\eta-1\,l} = partOf_\eta(h_{\eta i})$ **then** {received from all its parts}
17: **for all** $I \in \{1, \ldots, \mid D_{\eta i} \mid\}$ **do**
18: $c \leftarrow$**constraint**$(d_{\eta iI} = E_{\eta i}(v_{\eta-1\alpha}))\{d_{\eta iI} \in D_{\eta i}\}$
19: call $OptAlgth_{\eta i}(c, BestUtilBody_{\eta i}, I)$
20: **end for**
21: **end if**
22: **return**
23:
24: UTIL_msg_handler($BestUtil$, ind){by holon $h_{\eta-1\,k}$}
25: $h_{\eta\,i} \leftarrow superholonOf_{\eta-1}(h_{\eta-1k})$
26: $BestUtil_{\eta i}[ind] \leftarrow BestUtil$
27: **if** UTIL_msg received for all $ind \in \{1, \ldots, \mid D_{\eta i} \mid\}$ **then**
28: **if** $\eta < \eta_{max}$ **then**
29: send READY_msg($BestUtil_{\eta i}\, i$) to $headOfPart(h_{\eta i})$ {to its head}
30: **else** {Global Holon}
31: **Phase 2: Global Optimization**
32: $ind^* \leftarrow argmax_{ind}(BestUtil_{\eta i}[ind])$
33: $OptimumUtil \leftarrow BestUtil_{\eta i}[ind^*]$
34: $v^*_{\eta i} \leftarrow d_{\eta i\, ind^*}$
35: **Phase 3: Termination**
36: **for all** $h_{\eta-1\,k} = partOf_\eta(h_{\eta i})$ **do**
37: send VALUE_msg(ind^*) to $h_{\eta-1\,k}$
38: **end for**
39: **end if**
40: **end if**
41: **return**
42:
43: SOLUTION_msg_handler($SolInd$, $UpperInd$){by holon $h_{\eta i}$}
44: $SolInd_{\eta i}[UpperInd] \leftarrow SolInd$
45: **return**
46:
47: VALUE_msg_handler($UpperInd^*$){by holon $h_{\eta i}$}
48: $ind^* \leftarrow SolInd_{\eta i}[UpperInd^*]$
49: $v^*_{\eta i} \leftarrow d_{\eta i\, ind^*}\{d_{\eta i\, ind^*} \in D_{\eta i}\}$
50: **if** $\eta > 0$ **then**
51: **for all** $h_{\eta-1k} = partOf_\eta(h_{\eta i})$ **do**
52: send VALUE_msg(ind^*) to $h_{\eta-1\,k}$
53: **end for**
54: **end if**
55: **return**

Algorithm 2 Optimization Algorithm of holon $h_{\eta i}$

1: $OptAlgth_{\eta i}(ctr_{\eta i ind},\ BestUtilBody_{\eta i},\ ind)$
Require: - n-ary constraint $ctr_{\eta i ind}$
- $BestUtilBody_{\eta i}[k][l]$ corresponding to each $d_{\eta-1k\,l} \in D_{\eta-1\,k}$ associated with each subholon $h_{\eta-1\,k}$ of $h_{\eta i}$
Ensure: - the maximum $BestUtil$ corresponding to value $d_{\eta i ind} \in D_{\eta i}$ such that $BestUtil \leftarrow argmax_l(\sum_{j=1}^{M_{\eta i}} f_{\eta ij}(d_{\eta-1\,k\,l}) + +\sum_k BestUtilBody_{\eta i}[k][l])$
- the index $SolInd_{\eta-1\,k}$ of each variable $v_{\eta-1\,k}$ of the subholons of $h_{\eta\,i}$, corresponding to the solution.
2: executes the optimization algorithm of holon $h_{\eta\,i}$
3: send UTIL_msg($BestUtil$, ind) to $headOf_\eta(h_{\eta\,i})$
4: **for all** $h_{\eta-1\,k}$ = partOf($h_{\eta\,i}$) **do**
5: send SOLUTION_msg($SolInd_{\eta-1\,k}$, ind) to $h_{\eta-1\,k}$
6: **end for**
7: **return**

ing its variable domain, to all its parts via a VALUE message. By using that index, each part subholon will determine its own solution value, and recursively will send its respective index to its parts by VALUE messages. This phase is a top-down process, which is initiated by the global holon, propagates downwards down to all the atomic agents, when the meta-algorithm terminates. Figure 2 outlines HCOMA. The proof that it is sound and complete can be obtained in a straightforward way, as it will be shown next.

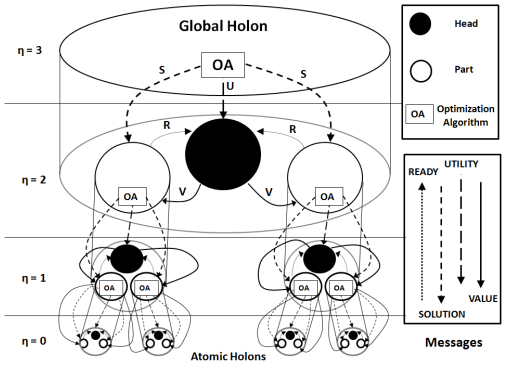


Figure 2: Outline of HCOMA

Proof of Correctness and Complexity
In any holon of any echelon, its *head* receives a UTIL message for each value of its variable domain. Each of these utility values is obtained from the execution of the optimization algorithm of the superholon, which uses the best utilities assigned to the domain values of the variables of each *part*. Since it is assumed that the distributed or centralized optimization algorithm is correct and terminates properly, if the best utilities associated with the *parts* domain values are correct, it can be concluded that the *head* will receive the best utilities for each value of its variable domain. But the atomic holons of echelon $\eta = 0$ own the best utilities for each value of their domains, for they depend only on themselves or internal optimization algorithms that are correct. Hence, by induction, it can be infered that any holon of any echelon will receive the best utilities for each value of its domain, since there is no *multipart* agent, i.e., there are no cycles in the graph made up of the holons and the constraints between them, and therefore the utility associated with each holon is considered only once. Thus, the global holon will choose the best utility for the entire holarchy, and then all the subholons will be informed and choose its respective domain values associated with that solution.

Due to its tree structure HCOMA has polynomial time complexity. As to the number of messages, it inherits the behavior of DTREE algorithm, which is linear in the number of agents (here holons) [Petcu and Faltings, 2004].

6 Modeling Example

As mentioned in section 4, the oil supply chain management is a real candidate problem to be modeled as a HCOP. In [Marcellino and Sichman, 2010] this problem was modeled as a DCOP, where the objective function is the total profit in the whole chain; it is also used a holonic approach: some holons, like the Transport Planners and the Derivative Producers, wrapped centralized optimizers.

In all the levels (echelons) of the supply chain the *logistics* is the *head* of each holon i.e., the *internal logistics* for the holon *refinery* or *terminal*, the *regional logistics* for the holon *region*, and so on, up to the *global logistics* for the holon *global*. Each *logistics* is responsible for the balance of products between the *suppliers* and the *clients*, and manages the transportation planning. The *supplier* role refers to any entity that provides products to other entity of the chain, such as a *refinery,* whereas the *client* role is played by any entity which needs these products, such as a *terminal*. The difference between availabilty and need of a product represents the emergent variable of the corresponding holon.

A refinery can produce multiple derivatives, and it does so according to different production plans, which are characterized by processing a definite quantity of a particular type of crude oil and producing a certain quantity of each resulting derivative. In addition, each refinery or terminal is responsible for the management of its inventories of each product. Thus, the decision variables of the model in the basic echelon ($\eta = 0$) are the *production plan* adopted by each refinery during each period of time, and the *inventory level* of each product in each refinery or terminal at the end of each period of time.

The higher the echelon, the larger the spatial scope of the corresponding holons. Similarly the higher the echelon, the longer the period of time considered by the head logistics in its planning. Thus, the holons result from a spatial and temporal discretization along growing abstract levels. On the other hand, the problem comprises different OPs: the production optimization of each refinery, and the transport optimization of the logistics in each holon. These latter are associated with growing echelons, and gradually embody larger geographic areas and longer planning periods of time. In fact, the inter-

	Ref. 1	Ref. 2	Income	Cost	Profit
Local Optim.	Plan B	Plan B	5920	629	5291
Holonic Model	Plan C	Plan C	10080	4162	5918

Table 1: Integrated holonic X Conventional approach

nal logistics is responsible only for a refinery or terminal on a day-by-day basis, the regional logistics takes care of an entire region with the week as the time unit, and so on up to the global logistics which focuses on the whole enterprise with a planning horizon of semesters or even years.

Let us consider a case study, which is simple but representative. It includes all the relevant entities of the chain and a significant set of products. It contains one continent with three regions, and one overseas SM. The first two regions comprise one refinery and one terminal, whereas the third region contains only one oil extraction area. Inside the regions, entities are connected by pipelines, but regions and SM are connected to each other by ships. The refineries produce three derivatives (gasoline, diesel and naphtha) by processing three types of crude oil (pet1, pet2 and pet3). The refineries can operate according to three production plans, which are specific to each refinery: plan A, plan B, and plan C. Although it is not a real situation, it is representative and fits for a proof-of-concept, which is accomplished by comparing the holonic model with a usual approach to manage the oil supply chain, which is based only on local optimization. The results of applying both approaches are presented in Table 1.

The local optimization approach recommends plan B in refineries 1 and 2, since it leads to the highest supposed profit (5291). On the other hand, the holonic model presents as optimum choice to adopt production plan C in both refineries, with a total profit of 5918. Such a discrepancy comes from the myopia of the local approach, which is unable to consider aspects of higher echelons, such as the additional profit resulting from sales of surplus products to SM, or the penalties incurred by not having product enough to supply all customer demands. Therefore, the proposed model generates a global gain of about 12 % on account of the whole chain integration.

7 Conclusions and Future Work

In this paper, we have defined a Holonic Constraint Optimization Problem (HCOP), which combines the distributed optimization constraint approach with the holonic multi-agent approach to take advantage of the best of both worlds. On one hand, since the constraint model provides a tight integration of the involved entities, it allows optimization. On the other hand, the holonic approach makes it possible to represent the intrinsic recursive nature of a category of optimization problems, such as the supply chain management. In addition, it was developed the meta-algorithm HCOMA for the solution of the HCOP. Since it is a meta-algorithm, it makes it possible to integrate different optimization algorithms, which may be chosen according to each specific problem.

In a future work the model will be extended to include more complex holarchies with multipart agents. Furthermore, it will be treated the environmental parameters and their influence on the stability of the holonic solution. In other words, it will be studied how an environmental perturbation propagates between echelons, and the possible advantages of the proposed model to tackle such kind of changes. Another point to be investigated is how the communication problems between neighboring echelons may harm the quality of a HCOP solution. Finally, it will be developed a prototype based on a case study of the oil supply chain, where the HCOMA will be implemented using as components centralized optimization algorithms already available in the oil industry.

Acknowledgment

Jaime Simão Sichman is partially supported by CNPq/Brazil.

References

[Gerber *et al.*, 1999] C. Gerber, J. Siekmann, and G. Vierke. Holonic multi-agent systems. *Research Report*, 99(3), 1999.

[Hilaire *et al.*, 2008] V. Hilaire, A. Koukam, and S. Rodriguez. An Adaptative Agent Architecture for Holonic Multi-Agent Systems. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1), 2008.

[Koestler, 1967] A. Koestler. *The Ghost in the Machine*. Hutchinson 'I&' Co, London, 1st edition, 1967.

[Mailler and Lesser, 2004] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445. IEEE Computer Society, 2004.

[Marcellino and Sichman, 2010] F. J. M. Marcellino and J. S. Sichman. Oil industry supply chain management as a holonic agent based distributed constraint optimization problem. In *AILog - ECAI 2010*, Lisbon, Portugal, 2010.

[Modi *et al.*, 2006] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2006.

[Petcu and Faltings, 2004] A. Petcu and B. Faltings. A distributed, complete method for multi-agent constraint optimization. In *Proceedings of the Fifth International Workshop on Distributed Constraint Reasoning (DCR2004) in CP 2004*, 2004.

[Petcu and Faltings, 2005] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.

[Petcu *et al.*, 2007] A. Petcu, B. Faltings, and R. Mailler. Pc-dpop: a new partial centralization algorithm for distributed optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 167–172, Hyderabad, India, 2007.

[Yokoo *et al.*, 1992] M. C. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *International Conference on Distributed Computing Systems*, pages 614–621, 1992.

Flexible routing combing Constraint Programming, Large Neighbourhood Search, and Feature-based Insertion

Philip Kilby[1,2] **and Andrew Verden** [1]
[1] NICTA, [2] Australian National University
Philip.Kilby@nicta.com.au, Andrew.Verden@nicta.com.au

Abstract

Vehicle Routing Problems in the Operations Research and Artificial Intelligence literature often allow only standard constraints plus (optionally) one "side" constraint. However, in practice, every problem has a number of side constraints, some of which have never been seen in the literature. This paper describes an architecture for handling such arbitrary side constraints, based on Constraint Programming, Large Neighbourhood Search, and sophisticated insertion methods. This architecture allows many problems that arise in fleet logistics to be solved efficiently.

1 Introduction

In the classical Vehicle Routing Problem (VRP), a set of customers must be visited by a fleet of vehicles at minimum cost. Typically, a constraint on the capacity of each vehicle is observed. In the VRPTW, an additional constraint forcing the visit time to fall within a given time window is also enforced. For surveys on methods for these classical problems, see [Toth and Vigo, 2002; Marinakis and Migdalas, 2007].

However, in many problems in logistics, while the basic structure of delivery by vehicles to locations is the same, problems exhibit a wide variety of additional constraints. These constraints can be fairly general - for example a constraint on vehicle re-use that allows a vehicle to perform subsequent routes. Some constraints, however, are very specific to a particular workplace, and not likely to be see again (for example overtime allowed only if no more than 5 days worked in the last seven, and also 6 hours break between the previous overtime shift and the last shift where a B-double vehicle was driven).

In this paper we examine methods which allow a wide variety of side constraints to be handled simultaneously. The system uses a combination of Constraint Programming (CP) and Operations Research (OR) techniques to achieve an efficient and flexible solution technique.

We begin with a description of the types of problem we wish to solve. We then describe an architecture that can handle these types of problem efficiently. We then suggest a solution technique based on Large Neighbourhood Search ([Shaw, 1997]) and insertion methods [Solomon and Desrosiers, 1988]. Insertion methods build a solution by inserting one visit at a time into an emerging route set. Although some insertion methods can give good results over a wide variety of problems, we show how bespoke insert methods offer a modular way of tailoring the solution method to speed up solution of problems with particular side constraints.

2 Problem Description

We wish to supply goods to n customers. Each customer order i specifies the location for service (l_i), and the quantity of goods required (q_i). A fleet of m vehicles is available to perform deliveries. Vehicle k has capacity Q_k. We are given the cost of travel between each pair of customers (c_{ij}).

In the most basic form of the problem, we wish to find a set of routes, one for each vehicle, such that customer requests are satisfied at minimum total cost, subject to the constraint that the total quantity of deliveries assigned to each truck does not exceed the truck capacity.

In this paper we use the term *route* to mean the sequence of customer *visits* performed by a truck.

This sort of formulation can represent problems where goods are delivered, or all goods are picked up, or problems where a service is provided.

Some additional problem variants and constraints have been studied in the Operations Research and Artificial Intelligence literature

- Time Windows, where the allowed times for the visit (earliest start time, latest start time) are specified (See [Bräysy and Gendreau, 2005] for a recent survey).
- Request/Request compatibility constraints specify that some pairs of visits cannot be assigned to the same route, or that they *must* be assigned to the same route. For example, it may not be permitted to carry particular chemical together on the same vehicle.
- Request/Vehicle compatibility constraints are similar, but they specify that some requests *must* be carried by a particular vehicle or *must not* be carried by a particular vehicle. [Nakari *et al.*, 2007] discusses problems with compatibility constraints.
- Pickup and Deliver Problem (PDP) constraints. Problems where goods are picked up at one location, and delivered to another. The PDP constraints ensure the

goods are picked up before they are delivered, and that the same route picks up and delivers. See [Savelsbergh and Sol, 1995] for a survey of PDP type problem and solution methods.

These basic constraints together also define the *General Vehicle Routing Problem* [Goel and Gruhn, 2008]. However, we wish to be able to solve problems with even more general constraints. Constraints such as the following have been investigated individually in the literature

- "Blood bank" constraint – a pickup and deliver where the pickup can occur any time in the morning, but delivery must be within 20 minutes of pickup. Because the delivery time window cannot be specified a-priori, this type of constraint cannot be expressed by the usual time window constraint.
- Multi-delivery (split-delivery) routing. Here, customer quantities may exceed the size of the largest truck, and so must be visited more than once. This type of problem is common in line-haul routing, where deliveries are made from a production facility to distribution centres (e.g. [Archetti *et al.*, 2006]).
- Loading dock constraint – Vehicle dispatch is limited by the number of docks available for loading. Alternatively, in PDP problems where there are multiple deliveries to a single customer, the number of deliveries that can be performed simultaneously is limited by the number of docks.
- Movable partition constraint. In this problem, multiple commodities are carried simultaneously. While the total capacity of the vehicle is fixed, each time it is loaded a decision can be made as to how much capacity is given to each commodity.
- Prize collecting problems, where visits have different values, and we wish to maximise the value of visits assigned less travel cost. This may mean that some visits are left unassigned ([Feillet *et al.*, 2005] discusses the case where there is a single vehicle).
- Route rendezvous constraints – that ensure different routes rendezvous to transfer goods. The time of the rendezvous may depend on the duration of individual routes.

These constraints are usually examined individually – that is, there is the usual vehicle routing problem plus one set of extra constraints that are known *a-priori*.

Our aim is to solve problems with a variety of these extra constraints applying simultaneously, while at the same time to avoid making the solution method dependent on which constraints are actually present.

The NICTA Intelligent Fleet Logistics project has developed a system called *Indigo* that is able to handle a variety of problems in logistics. Its architecture and methods are described in the following sections.

3 Architecture

Constraint Programming is an obvious technique to express the side constraints seen in practical Vehicle Routing problems, and to support solving instances of these more general problems.

However, constraint programming can introduce an expensive overhead to handle some constraints. For instance, if a capacity constraint is tied to the variable which indicates the route to which the visit is assigned, then each time that variable is altered, or any time the load changes on any route which is within the domain of that variable, then the capacity constraint will be re-checked. This will ensure correct operation, but can lead to much redundant checking.

For this reason, the *Indigo* system has two "classes" of constraint.

The first class of constraints (called "native" in this context) include all the constraints of the General Vehicle Routing Problem [Goel and Gruhn, 2008]:

- Capacity constraints (across multiple commodities but with fixed capacity for each commodity)
- Usage constraints, that limit the total usage of resources such as time and distance accumulated during each run.
- Time window constraints, specifying earliest and latest start time.
- Pickup-and-Deliver constraints, enforcing precedence and same-route constraints between a pair of visits.
- Request/Request and Request/Vehicle compatibility constraints

Native constraints are handled very efficiently by the system with minimal interaction with the Constraint Programming system. Any decision made by the solver is guaranteed to observe all of the native constraints.

However, additional side constraints can be specified and handled using the CP system. In the long-term, we wish to be able to use the Zinc language [Marriott *et al.*, 2008] to specify these constraints, and then solve the problem within the G12 system [Wallace and the G12 Team, 2009]. In the short term, however, propagators for each side-constraint are hand coded, using a simple bespoke constraint programming system.

Even with some hand-coding still required, the advantage of using a CP paradigm is clear. The propagator for each constraint is a self-contained piece of code that is only "known" to the CP system. The alternative in traditional OR systems would require the main body of VRP solving code to be modified for each new constraint, which makes maintenance difficult, and unexpected interactions almost inevitable.

Special, separate modules also convert different variants of the problem into standard form. For multi-delivery routing, for instance, a separate module breaks each order quantity into smaller pieces that can be assigned efficiently to trucks of various sizes.

The advantage of approach handling *native* constraints internally was demonstrated in [Kilby *et al.*, 2010]. It was shown that handling these constraints internally, rather than as constraints in the CP system, decreases the CPU time by a factor of about 2. It also slightly improves solution quality in some cases.

4 Interaction with the CP System

The *Indigo* system is integrated with a CP system. The CP system has a number of variables for each visit and route, including: A *successor variable* indicating which visit should follow the given visit. A *predecessor variable* indicating which visit should precede the given visit. A *route variable* indicating which route the visit is assigned to. If time is used, then *arrive time* and *service start time* variables are used. For each assigned visit, and for each commodity, there are variables specifying the cumulative load.

Constraints can be posted in the CP system to limit the values these variables can take. For instance, in the case of the blood donor constraint, as soon as the pickup visit is assigned, then the delivery visit will have its service start time variable constrained.

In operation, the *Indigo* system acts as a variable/value choice heuristic for the underlying CP system. *Indigo* maintains an internal representation of an emerging route set, including the list of partially built routes, plus the list of yet-to-be assigned visits.

Each time *Indigo* is called, it chooses a visit to insert, and a position in which to insert it. It can then propagate the effects of this choice to the CP system.

As discussed in [Kilby and Shaw, 2006], chronological backtracking in CP imposes limits on the amount of information that can be propagated to the CP system. For instance if the *Indigo* system decides to place visit 10 after visit 2, we cannot bind the *successor* variable of visit 2 just yet. If we were to make the assignment *succ[2] = 10*, then we would not be able to insert any other visit after 2 for the rest of the execution of the procedure. So instead, we make all propagations that can be inferred from the assignment. For instance, we can remove 10 from the successor variable of every visit except 2. We can also update the route variable for visit 10. A number of other propagations are possible. For example, let us say that visit v will follow visit p in route r

- All other assigned visits (except p) can be removed from the predecessor variable of v. Similarly the successor variable of v can be updated.
- If we assume the triangle inequality for time ($\forall a, b, c, t_{ac} \leq t_{ab} + t_{bc}$) then we know that we cannot arrive at v any earlier than we currently do. We can therefore update the bound on the arrival time to be at most the current arrival time.
- The latest arrival time cannot be any later than the current value (again assuming the triangle inequality). Hence we can also update the upper bound on the arrive time to be the current latest feasible arrival time.
- In pickup-up only problems, the load on any commodity cannot be less than the current value. We may therefor update the appropriate bound on the load variable.

If, during execution of the propagations following an assignment, a failure occurs (i.e. CP has identified an inconsistency) then the internal data structure within *Indigo* must be updated as part of the backtracking of the CP system. The CP system will ensure that the same assignment is not attempted again in the future.

When the *Indigo* system is subsequently called, changes to the successor, predecessor or time variables must be noted, and the next choices must be consistent with these values.

5 Solution method

Like many VRP solution methods, *Indigo* relies on local search methods to improve an initial solution. Many local search techniques for routing problems have been developed (for example 2-opt, 3-opt, Or-opt). However, these methods that move directly from one solution to another do not make use of the full power of CP.

Again, local search methods are limited by the chronological backtracking restrictions imposed by the CP architecture. Hence, methods that build up a solution one piece at a time, using CP search procedures, are preferable to local search type methods such as those above that move directly from solution to solution.

Large Neighbourhood Search (LNS) [Shaw, 1997] is a local search procedure where part of a solution is destroyed, and a then a new solution created by finding new values for the freed variables. This method uses exactly the sort of incremental solution building method that can exploit propagation in CP to guide the solution towards good, feasible solutions.

The *Indigo* system draws on the work of Ropke and Pisinger [Ropke and Pisinger, 2006] (R&Phere). Like that work, it uses insertion methods to create an initial solution, and then again to repair the solution in each iteration of LNS. The LNS algorithm can be given as follows:

1 Create initial solution S

2 Choose a "destroy" method d

3 Create S' by removing customers from S according to method d

4 Choose an insert method i

5 Create solution S'' from S' by inserting customers according to method i

6 If the acceptance method accepts solution S''

7 Replace S with S''

8 If iterations remain, return to line 2

This method is characterised by

- The destroy methods available at line 2
- The insert methods available at line 4
- The acceptance methods available at line 6
- The number of iterations available at line 8.

In this paper, we look only at enhancements to the insert methods available at line 4 that allow for some of the more general instances to be solved effectively. These methods are described in the next section.

6 Insert methods

Insertion methods proceed by repeating two stages:

First, amongst all unassigned visits, the best position to insert each is selected. Then, the visit which is to be inserted is chosen. The visit is then inserted in its best position. Solomon

[Solomon, 1987] seems to be the first to suggest this two-score system.

The best insert position for each still-unassigned visit is then updated. The method can then iterate until all visits have been assigned a position.

In much previous work, only a limited number of features are considered when making these two choices. In some work, only *minimum cost insertion* is considered – i.e. visits are always inserted in the position which gives rise to the smallest increase in cost, and the visit with the smallest increase is inserted first.

R&Pshow that using a combination of insertion methods gives better results than a single method, as it allows different methods to be used at different times. Running on benchmark problems with limited constraints (PDP, time window and capacity constraints), R&Pidentified a set of insert methods that gave good performance.

We wish to extend this idea, and use a variety of insertion criteria when making these choices. We will show below how this can be advantageous in real-world problems.

We will describe several criteria, or "features" which can be used in either choosing where to insert a visit, or choosing the visit to insert. Each criteria is described below. The degree to which a particular feature is present is rated on a score of 0 to 1, with 0 meaning "not present", and 1, "present". Along with each feature, the "base" value which is used to normalise the value (as described in Section 6.1) is also given. If *reversed* is specified, then (1 - val) is returned, rather than val. Two types of normalisation are also possible, as discussed in Section 6.2

The following symbols are used in the description below. The visit v is to be inserted between p and s on vehicle k. The cost of insertion is $c' = c(p,v) + c(v,s) - c(p,s)$.

Route domain Favour visits with few feasible routes. Val is number of routes v can be feasibly inserted into. Base is total number of routes. Reversed.

Num ins pos Favour visits with few feasible insert positions. Val is the number of feasible insert positions. Base is number of assigned visits. Reversed

Distance to depot Favour visits far from a depot. Val is distance to the closest route start or end. Base is max dist to route start or end.

Value For use in prize-collecting problems, favours inserting high-value visits first. Value is prize-value of the visit. Base is max prize-value over all visits.

Load Favour largest load first. Value is load. Base is max vehicle capacity

Nearest neighbour Encourages v to be inserted near its neighbours. Val is min $(c(p,v), c(v,s)$. Base is distance to v's 10^{th}-nearest neighbour. Reversed. Normalised with method 2.

Min insert cost Cheapest insert first. Value is c'. Base is twice the average insert cost. Ave insert cost is (Total cost of inserted visits) / (number of inserted visits). Reversed. Normalised by method 2.

Max insert cost Reverse of above. Calculated same way, but not reversed.

Regret, 3-Regret, 4-Regret See below. Base is same as *Minimum insert cost*. Normalise by method 2.

Rand Randomise slightly. Val is a uniform-random number in [0,1).

Time Window width Encourages smallest time window to be inserted first. Val is width of v's time window. Base is max of time window widths. Reversed.

Time Window end Encourages visit with latest time window to be inserted first. Val is the end time of the last time window. Base is max time window end.

Wait time Encourages vehicles not to arrive at a location before the start time window (as the vehicle must then wait for the time window to open). Val is the time the vehicle must wait at v before service starts. Base is (Last time window) / 10. Reversed.

Pickup Late, Deliver early Encourages vehicle to do deliveries at a location before doing pickups.

Lost slack Encourages spare time to be preserved. Val is how much "spare time" (difference between arrival time and time window end) is lost at s. Base is max time window width. Reversed. Normalised using method 2

Fill vehicle Used when problem has a bin-packing flavour, and favours inserts that fill the vehicle. Value is spare capacity after insert. Base is max capacity. Reversed.

Balance routes Encourages routes to have similar length, as measured by difference between shortest and longest route. Penalise adding to longest route, and reward adding to shortest.

6.1 Base values

Base values are used to normalise the feature values into the range [0,1] by diving *val* by *base*. Since we wish the values to be comparable, we must be careful in the base value chosen. For example, for nearest neighbour, the "safe" base is the length of the longest arc in the problem. However, this is likely to be very large, and not ever used in a solution. We therefor use a base value which is the distance to the 10th-nearest neighbour, as the neighbour of most visits is likely to come from this set.

6.2 Normalisation

The basic method of normalisation is to simply divide by the base. This is done whenever the base is a guaranteed maximum for the feature value (e.g. maximum time window width as base for the time window width feature)

However, for reasons outlined above, some base values are "optimistic" or heuristic values, and can be exceeded. Since we still wish to rank values that are greater than the base value, an alternative, non-linear normalisation is used. To normalise a value v with a base b, the normalised value is calculated as follows

$$\text{tmp} = \max(v/b, 0); \text{return tmp}/(0.5 + \text{tmp});$$

This normalisation always falls in the range [0,1]. The values 0 to 1 map to normalised values 0 to 0.6667, with 0.5 mapping to 0.5. The value of 0.6667 makes values normalised using this method approximately comparable to values normalised with the first method.

6.3 Regret

Regret is based on the difference between the best and next-best insert positions for a visit. If there is a big difference (a large regret), then if the visit does not get its favoured position, the effect on the objective is high. The method therefore chooses the visit with maximum regret to insert first.

More formally, if the minimum cost to insert visit i in route k is c_{ik}, and the permutation $o(k)$ permutes the routes into increasing order; i.e. $c_{i,o(1)} \leq c_{i,o(2)} \leq ... \leq c_{i,o(m)}$. Then regret($i$) is

$$c_{i,o(2)} - c_{i,o(1)}$$

3-Regret allows slightly more look-ahead, taking the first three positions into account: Then 3regret(i) is

$$((c_{i,o(2)} - c_{i,o(1)}) + (c_{i,o(3)} - c_{i,o(1)}))/2$$

4-regret is defined analogously, over the cost of the four cheapest routes.

6.4 Implementation

All of these methods (except the regret methods) can be calculated using just the visit to be inserted and the predecessor in the route. In the *Indigo* system, features are defined using a base class. New features can be incorporated easily by specialising the base class to calculate the required value.

7 Use of features

Features are combined using weights. Two separate weight sets are used – weight set 1 is used to decide which visit to insert; weight set 2 is used to decide where to insert it. Each score is simply the scalar product of the weights and the feature values.

For example, selecting the visit to insert using 3regret with a small amount of randomness; and position to insert using min-insert-cost, the following could be used
Feature set 1: { 3regret, rand }. Weight set 1: { 0.95, 0.05 }.
Feature set 2: { min-insert-cost }. Weight set 2: { 1.0 }.

For efficiency, only those features with a non-zero weight need to be evaluated. Weight sets can be general-purpose, or specific for a a portfolio of instances.

The new insert features allow flexibility in non-standard problems. For instance, in a multi-delivery pickup-and-delivery problem there may be multiple pick-ups and deliveries at a single location. While this type of problem is seen relatively often in practice (it is one way of modelling a vehicle leaving and returning to the depot multiple times) it does not appear in benchmarks. There is nothing in a standard VRP heuristic which makes us deliver before we pick up at the same location. The "pickup-early, deliver-late" ensures this sequence. However, in standard benchmarks, there is no call to use such a feature. It is only in the more flexible routing that it is useful – but there it is indispensable.

8 Computational testing

In order to test the effectiveness of the architecture, the system was tested on some standard benchmarks. These do not exhibit the flexibility of the system, but indicate the effectiveness on standard problems.

The system was tested on the VRPTW benchmark problems of Solomon [Solomon and Desrosiers, 1988] with 100 customers, and the extended Solomon benchmarks of Gehring and Homberger [Gehring and Homberger, 1999] with between 200 and 1000 customers.

The experimental setup used "standard" parameters similar to those used in R&P

- Accept function is Simulated with a temperature gradient of 0.99975, and an initial probability chosen so there is a 50% chance of accepting an increase of 5%.
- Removal selection functions and Insertion functions as per R&P.
- Adaptive learning of which selection and which insertion method to use, using rewards similar to those used by R&P
- 30,000 iterations of LNS
- 50 customers removed for size 100 and 200 problems. 100 customers removed for larger problems.
- 5 runs of each problem, best solution reported

Because the current *Indigo* system does not have a feature to reduce the number of vehicles, the problems were modified so that only the number of vehicles in the best-known solution were available to the system. This makes the comparison a little less fair, as most systems initially try to reduce the number of vehicles. However, it is consistent with many real-world problems where the vehicle fleet is fixed *a-priori*.

Because of limited space to report we give just the basic results. We express performance as a ratio of the increase as a ratio of best-known solution. E.g. 1.02 means the results was 2% higher than the best-known solution.

All problems were solved with the best-known number of vehicles. We produced new, best-known solutions to 83 of the 300 benchmarks. Table 1 shows the results. *Size* gives the number of customers in the problems; *Best* gives the number of problems where a "new best" solution was found; *Mean* is the mean increase; *80%* gives the 80th percentile of increase (i,e, 80% of values were less than this value); and *Max* gives the maximum increase over best-known solution. *CPU* gives mean time per run in CPU seconds. The tests used one CPU of an 8-core 32 bit Intel Xeon running at 2GHz.

Source	Size	Best	Mean	80%	Max	CPU
Solomon	100	0	1.01	1.01	1.05	53
G & H	200	11	1.01	1.02	1.05	120
G & H	400	13	1.01	1.03	1.06	487
G & H	600	19	1.02	1.04	1.10	766
G & H	800	18	1.02	1.05	1.11	1108
G & H	1000	22	1.03	1.06	1.14	1450

Table 1: Results on benchmark problems

For the smaller problems (100-200 customers) these results are very good for relatively small CPU times. Since other systems often run for much longer than the maximum 30 minutes allowed this system, the results for larger problems are reasonable, although some work is required to ensure the system performs as well on the larger problems as it does on the small.

9 Future work

With a large number of features, the space of possible weights is very large. The task of finding effective feature weights can be very difficult. We are currently looking a two methods for choosing weights: *static* and *dynamic*. Static methods will calculate a weight set *a-priori*, using a portfolio of similar problems from a given user. Dynamic method we make use of the fact that in LNS we are essentially solving the same problem many thousands of times. We can dynamically adapt the weight set, and test the effectiveness of the new set in subsequent runs.

We are also looking at ways of increasing the search diversity in larger problems, to improve the performance on some of the larger problems reported in section 8.

10 Conclusions

We have described an architecture for solving a variety of logistics problems, including, for instance, line-haul problems that are not well suited to traditional VRP methods. This architecture has the advantage of handling many of the most common constraints very efficiently, while allowing additional side constraints to be specified and handled in a modular and flexible way by an underlying Constraint Programming system.

We have argued that sophisticated insertion methods make an ideal partner for Large Neighbourhood Search and Constraint Programming for solving real-world vehicle routing problems. We have given a method of calculating and combining a number of feature scores, that allows insertion methods to be tailored more easily to the characteristics of the problem at hand.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[Archetti *et al.*, 2006] C. Archetti, M. G. Speranza, and A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73, 2006.

[Bräysy and Gendreau, 2005] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[Feillet *et al.*, 2005] Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188, 2005.

[Gehring and Homberger, 1999] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In K. Miettinen, M. Makela, and J. Toivanen, editors, *Proceeding of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science,*, pages 57–64. University of Jyväskylä, 1999.

[Goel and Gruhn, 2008] Asvin Goel and Volker Gruhn. A general vehicle routing problem. *European Journal Of Operational Research*, 191(3):650–660, 2008.

[Harvey and Ginsberg, 1995] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, volume 1, pages 607–615, Montréal, Québec, Canada, 1995. Morgan Kaufmann.

[Kilby and Shaw, 2006] Philip Kilby and Paul Shaw. Vehicle routing. In F. Rossi, P. Van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, chapter 23, pages 801–836. Elsevier, 2006.

[Kilby *et al.*, 2010] Philip Kilby, Andrew Verden, and Lanbo Zheng. The cost of flexible routing. In *Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN) 2010*, 2010. To appear.

[Marinakis and Migdalas, 2007] Yannis Marinakis and Athanasios Migdalas. Annotated bibliography in vehicle routing. *Operational Research*, 7(1):27–46, 2007.

[Marriott *et al.*, 2008] Kim Marriott, Nicholas Nethercote, Reza Rafeh, Peter J. Stuckey, María García de la Banda, and Mark Wallace. The design of the Zinc modelling language. *Constraints*, 13(3):229–267, September 2008.

[Nakari *et al.*, 2007] Pentti Nakari, Olli Bräysy, and Wout Dullaert. Communal transportation: Challenges for large-scale routing heuristics. Reports of the Department of Mathematical Information Technology Series B. Scientific Computing B6/2007, University of Jyväskylä, 2007.

[Ropke and Pisinger, 2006] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[Savelsbergh and Sol, 1995] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–39, 1995.

[Shaw, 1997] Paul Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Working paper, University of Strathclyde, Glasgow, Scotland, 1997.

[Solomon and Desrosiers, 1988] Marius M. Solomon and Jacques Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–12, February 1988.

[Solomon, 1987] M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35:254–265, 1987.

[Toth and Vigo, 2002] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, PA, 2002.

[Wallace and the G12 Team, 2009] Mark Wallace and the G12 Team. G12 – towards the separation of problem modelling and problem solving. In *Proc. CP-AI-OR'09*, volume 5547 of *Lecture Notes in Computer Science*, pages 8–10. Springer, 2009.

Optimising Efficiency in Part-Load Transportation*

Srinivasa Ragavan Devanatthan, Stefan Glaser and **Klaus Dorer**
Hochschule Offenburg, Offenburg, Germany
sdevanat@stud.hs-offenburg.de
{Stefan.Glaser, Klaus.Dorer}@hs-offenburg.de

Abstract

Existing approaches solving multi-vehicle pickup and delivery problems with soft time windows typically use common benchmark sets to verify their performance. However, there is a gap from these benchmark sets to real world problems with respect to instance size and problem complexity. In this paper we show that a combination of existing approaches together with improved heuristics is able to deal with the instance sizes and complexity of real world problems. The cost savings potential of the heuristics is compared to human dispatching plans generated from the data of a European carrier.

1 Introduction

With an increase in transport business and many mergers between major logistics companies, it becomes increasingly difficult for the the dispatchers to have an overview of the orders relevant to their business. Consequently, opportunities to load orders together on the same vehicle are missed frequently, resulting in increased costs.

Many optimisation algorithms fail to improve the situation due to the size of the problem instances and the complexity of the constraints involved. In this paper we show that a combination of existing approaches together with improved heuristics is able to deal with the instance sizes of real world problems and reduce the costs of transport plans considerably. To do so we have used real data of a major logistics carrier and compared the results of our approach with the transport plan that has been created by human dispatchers and has been performed by the vehicle fleet.

The rest of the paper is organised as follows: Section 2 introduces the transport domain. Section 3 explains how real world problem instances can be addressed with results shown in section 4 before we conclude in Section 5.

2 Domain

The multi-vehicle pickup and delivery problem with soft time windows (m-PDPSTW) [Psaraftis, 1995; Dorer and Calisti, 2005] consists of finding optimal plans for serving transportation requests of customers. The problem is 'single-vehicle' if all transportation requests are served by a unique vehicle. Here, we deal with a 'multi-vehicle' problem where multiple vehicles can be used for transporting all orders. The vehicles may be of different type and have different capacities. As opposed to vehicle routing problems [Laporte and Osman, 1995], in *pickup and delivery problems* (PDP), vehicles do not necessarily start or end in the same location. Transportation requests may have the same, but usually different, pickup and delivery locations. The pickup and delivery of orders has to occur within a specific time window, even though time constraints can be possibly violated up to some tolerated degree. These kind of problems are called PDP with soft time windows.

Table 1 presents different approaches from literature that directly handle PDPTW problems. It is worthy of note, that the approaches defined in the table dealt with benchmark instances while we report on results on significantly larger instance size from real world.

A rich overview on different versions of the problem as well as a collection of solution methods and applied heuristics can be found in [Parragh *et al.*, 2008a; 2008b].

In the following, we specify the information and con-

Method	Author(s)	Characteristics
Insertion heuristic	[Jaw *et al.*, 1983]	300 orders, 24 vehicles
Clustering followed by decomposition	[Dumas *et al.*, 1991]	880 orders, 53 vehicles
Reactive tabu search	[Nanry and Barnes, 2000]	100 orders, 10 vehicles, based on VRPTW instances of [Solomon, 2005]
Branch and cut algorithm	[Ropke *et al.*, 2007]	40 instances of [Savelsbergh and Solomon, 1998]
Insertion heuristic with k-opt	this paper	2137 orders with 1736 available vehicles

Table 1: Methods for m-PDPTW

*This work is supported by the *IngenieurNachwuchs* program of the German BMBF grant number 17 N25 09.

straints of the domain relevant for our work. The term *node* is used to indicate the combination of a stop location and the corresponding time (arrival and departure time) for a given vehicle. A *leg* is the path between two nodes. A *route* is the sequence of nodes a vehicle visits. The vehicle is assumed to be empty at the beginning and at the end of a route. The sum of all routes is called the *delivery plan* representing the schedule of each vehicle. The quality of the solution is the cost of the delivery plan (see Section 2.2).

2.1 Initial Information

The information required to solve transport optimisation is, a set of transport requests or orders and the set of vehicles that are available. Also information for the distance and drive time required for driving any possible leg has to be available.

Every *order* specifies: order type, capacity demand (loading meters), weight, pickup location, pickup time window, pickup service time, delivery location, delivery time window, delivery service time and the time at which the order is known to the system.

The *vehicle* definitions include: vehicle type, capacity (in loading meters and weight), availability location and time. A mathematical analysis of the specific data used is presented in chapter 4.

2.2 Cost Model

Cost reduction is a main driving factor for logistics companies. Cost is therefore used as the objective function for optimisation. The cost model has to make sure that solutions are preferred by the optimisation algorithm that are cheaper to perform in practice. It has therefore to reflect the real costs of the companies as close as possible.

Two types of cost models are typically distinguished: fix-variable for own vehicles and matrix-based for subcontracted vehicles. Costs for own vehicles of the fleet are calculated as

$$c_{fv} = c_{fix} + c_{var} \tag{1}$$

with $c_{fix} = k_{fix} * t$ and $c_{var} = d_{empty} * k_{empty} + d_{loaded} * k_{loaded}$. k_{fix} is a constant representing the fix costs per day. It may depend on the vehicle type in general, but did not in the context data of this paper. t is the number of days the route covers. d_{empty} is the sum of distance of all legs driven empty including a possible empty leg to the first pickup location and a possible empty leg to drive home at the end of the route. d_{loaded} is the sum of distance of all legs where at least one order is loaded. k_{empty} and k_{loaded} are costs per kilometre for empty or loaded legs respectively.

Costs for subcontracted or spot market vehicles are typically based on distance and load matrices.

$$c_{ma} = \sum_{i=1}^{n} d_i * l_i * k_{ma}(d, l) \tag{2}$$

where n is the number of legs, d_i is the distance of leg i, l_i is the load on leg i in loading meters and $k_{ma}(d, l)$ is a function defining the costs per kilometre and loading meter. The function is represented by a matrix defining different distance and load classes with linear interpolation between the specified values. It is usually retrieved from historic data. Note that this cost model does not account for fixed costs nor does it take empty legs to the first pickup or after the last delivery into account. Comparable entries of the cost constants in the matrix are therefore typically much higher than k_{empty} and k_{loaded}. In the context of this paper, two distance classes and thirteen load classes have been used.

2.3 Constraints

The optimisation heuristics have to obey certain constraints in order to create solutions that are drivable in reality.

- Load constraints:
  - Precedence (pickup has to be before delivery);
  - Pairing (pickup and delivery have to be done by the same vehicle);
  - Capacity limitation of a vehicle;
  - Weight limitation of a vehicle;
- Time constraints:
  - Earliest pickup and delivery;
  - Latest pickup and delivery;
  - Legal driving time regulations for drivers.
  - Service times at pickup/delivery locations

In practice pickup and delivery times are typically treated as soft constraints. This means that short delays are accepted if they allow for better delivery plans. A soft constraint is defined by a tuple $< s, e, c_f, c_v >$ where s is a start value above (below) which the condition is soft violated, e is an end value above (below) which the constraint is considered hard violated, c_f are the fix violation costs assigned if the constraint is (soft) violated and c_v are variable violation costs that grow proportional to the amount of soft violation. The fixed violation costs can be used to control the number of violations. The variable violation costs ensure that the amount of constraint violation is kept low and only accepted if the violation cost is less than the benefit of violating the constraint.

3 Implementation Strategies

The classification of the problem as NP-Hard and the size of the problem in reality, thousands of orders to be served with a fleet of hundreds of vehicles, impairs the application of exact methods. Most exact methods, which work well for small specific problem instances in the absence of many constraints, fail to work acceptably fast in practice. Heuristics find *good* solutions in reasonably short time, which is the major concern in the real world.

A straight-forward method to apply insertion heuristic to build an initial solution, followed by a tour improvement heuristics seems the first best tentative approach towards a problem of the size we have handled.

3.1 Insertion Heuristic

The insertion heuristic builds a set of routes by inserting one order at a time. The number of routes is freely determined while inserting. It is not expected that it produces the optimal set of routes for transporting the orders. The main idea is to build an initial feasible solution which is then optimised for the objective function. The quality of this initial solution

depends on the sequence in which the orders are inserted. In our case orders have been sorted by earliest delivery time.

A new order is inserted at the best feasible insertion place over every route. It considers the objective function of the problem as the insertion cost. For TSP problems, the objective function is *distance* and an example would be the *smallest detour in distance* [Azi *et al.*, 2010], which may not find the optimal tour, but would certainly produce an acceptably short route.

For the m-PDPSTW that we handle, each route in the solution is a TSP but with constraints which impose precedence of the pickup node before the delivery node. This is referred to as the *pickup and delivery*-TSP or PDTSP, where the objective function is the total cost.Here, we require a simple permutation heuristic which would find the cheapest point of insertion of a new order on the route. This heuristic, *cheapest permutation*, is described below.

Let $(i_0, \dots i_n)$ be the nodes on the route r. Let i_p and i_d be the pickup and delivery nodes of a new order that has to be inserted on the route. The pickup node i_p is inserted as $(i_{k-1}, i_p, i_k),\ 1 \leq k \leq n,\ i_0$ is the start node and i_{k-1} and i_k are two adjacent pickup or delivery nodes on the route. For each partially inserted route, $(i_0, \dots, i_p, i_k, \dots, i_n)$, the delivery node i_d is inserted as $(i_{l-1}, i_d, i_l),\ k \leq l \leq n$. If the pickup-insert fails, the following permutations of delivery-inserts are not made.

For each order, the insertion heuristic is run on every route and the cheapest route is chosen. If no existing vehicle is able to transport the order, a new vehicle with a new route is created to handle this order. The set of all routes serviced by individual vehicles constitutes a delivery plan and is an initial solution.

The way in which precedence constraints are incorporated during the solution process is of particular importance to the effectiveness of this heuristic for this problem. It implicitly eliminates some of the infeasible PDTSP solutions. The heuristic is fairly quick mainly because it does not permute the existing nodes on a route when an insertion is made. The constraints are enforced at different levels of the heuristic.

- It is possible that the load constraints of the vehicle are hard violated at node i_p. These are physical restrictions of the vehicle and cannot be soft violated. In such cases, the corresponding permutations of delivery-inserts are not made, reducing the feasible states.
- A pickup-insert could alter the time parameters of the subsequent nodes after i_p, which may produce a violation of the time constraints up to a certain limit. In these cases, the subsequent nodes are one of the permutations of the delivery-inserts. The heuristic first constructs the route and then schedules. If scheduling fails, the route is thrown away.

Soft constraint violations produce costs which are included in the objective function of the problem. This ensures that an order is allocated on a route with a soft violation only when allocating the same order on all other routes is impossible or produces a higher cost.

The driving plan may either be improved by applying restrictive constraints which enforce route quality at the time of construction or by using tour improvement heuristics. Improvement may be achieved by a re-arrangement of existing nodes in a route or the re-assignment of an order to another route.

3.2 Tour Improvement Heuristic

The tour improvement heuristic aims to improve the quality of the entire delivery plan by employing a local search with *k*-change neighbourhood, simply referred as *k*-opt [Papadimitriou and Steiglitz, 1982; Helsgaun, 2006]. It has been applied for the travelling salesman problem and has been shown to produce high quality solutions in polynomial time [Lawler *et al.*, 1985]. As far as we have seen, the k-opt has not been applied for the m-PDPSTW. In this Section, we define a single *"change"* for the m-PDPSTW and brief on the k-opt.

A new feasible solution can be obtained by performing a single change to an existing solution. If k' number of changes are applied to the current solution, the new solution is then described to be in the k'-neighbourhood. Depending on the type of the problem, the parameter of change can be varied. This is explained as follows.

Let $R = \{r_1, r_2, \dots r_n\}$ be the set of all routes. Each route is serviced by a single unique vehicle. Let $O_i = \{o_1, o_2, \dots o_k\}$ be the orders transported by route r_i. It should be noted that $O_{all} = \sum_{i=1}^{n} \mid O_i \mid$ and $O_i \cap O_j = \phi$ for distinct $i, j \in \{1, \dots n\}$. As can be seen, here we assume that each order is transported on a unique route. Additionally, each route is assumed to be serviced by a unique vehicle.

A single *change*, $k' = 1$, is then, removing an order from the route it is transported on and inserting the order on another route. This remove-insert pair is together considered as a single change. Therefore, a removal is always followed by an insert. It is possible that a route in the current solution might violate constraints beyond their hard limit, either at the time of removal or at the time of insertion of the order. In either case, the new solution is not accepted as an improvement. This ensures that all the orders transported in the current solution are also transported in the successor solution.

In the case of TSP, a k-opt move changes a tour by replacing k edges between existing nodes, with k other edges such that a shorter tour can be obtained. For the problem instance handled in this paper, a m-PDPSTW, k orders transported on one route is replaced with k other orders from a different route, such that the total cost is reduced. These k-changes can be sequential as well as non-sequential. If a k-change produces a better solution in terms of the objective function, then this new solution is accepted as the current solution for further improvement.

The pseudo-code for tour improvement is shown in Algorithm 1. Picking routes in lines 4 and 5 is done in a brute force approach iterating over all routes. As stop-criterion a time limit as well as a maximal number of iterations was used. Lines 7 to 13 perform a hill-climbing in the k'-neighbourhood (not including the $k' - 1$ neighbourhood).

In cases where the neighbourhood does not have feasible and cheaper solutions, the neighbourhood is enlarged i.e., $k' = k' + 1$ changes are made to search for an improvement (line 6). A k-opt heuristic checks all k' neighbourhoods before termination. Here, $1 \leq k' \leq k$. It can be observed

Algorithm 1 tour improvement

```
1: procedure IMPROVE(s, k)                ▷ initial solution, max neighbourhood size
2:     while true do
3:         d ← dimensionOfSearchSpace(s)
4:         pick route1 from d
5:         pick route2 from d
6:         for k' ← 1, k do
7:             o ← neighbourhood(route1, route2, k')
8:             for all o' ∈ o do
9:                 s' ← getNeighbour(o')
10:                if f(s') < f(s_best) then
11:                    s_best ← s'
12:                end if
13:            end for
14:            if stop-criterion met then
15:                return s_best
16:            end if
17:            s ← s_best
18:            if improvement then
19:                break                  ▷ repeat with k = 1
20:            end if
21:        end for
22:    end while
23: end procedure
```

that the k-opt may take exponential number of iterations to evaluate all possible k'-changes. The performance is hence sensitive to the number of orders on a route.

A similar k-opt algorithm for the TSP was studied for theoretical performance guarantee and results have been shown for the proof of quality of the 2-opt as a heuristic for random TSP instances in unit square [Chandra *et al.*, 1994]. The advantage of the k-opt heuristic is its scalability in the choice of the neighbourhood size. We show the results obtained from using the 2-opt as the *tour improvement heuristic*. Though k can take any integer value less than $\mid O_i \mid$ for that route r_i involved in the change, we applied $k = 2$. The restriction was not just to simplify implementation, but to see the first results of performance of the 2-opt algorithm on real world data for the m-PDPSTW, on an inexpensive hardware. The results are discussed in the following Section.

4 Results

The heuristics described in the previous section were evaluated on real world data of an international carrier. The availability of the human delivery plan allows us to compare the performance of the heuristics and demonstrate their applicability on real world problems. However, to have an idea on the gap of the used heuristics to optimal solutions, we ran them on the biggest available benchmark problems listed in [Lim, 2010]. In the following we start with a discussion of the benchmark problem data sets and the differences to real world scenarios, before we characterize the specific real world data set followed by a comparison of our results to the human delivery plan.

4.1 Benchmarks

In order to run benchmark problems, a couple of changes to the system are necessary. The expensive calculation of legal drive time regulations could be switched off. In general the drive time/distance lookup is much cheaper in the benchmark case just calculating Euclidean distances instead of looking up real road drive times. To avoid too heavy changes on the system, distance and drive time calculation is done using integers (rounded, not truncated) which is precise enough in real world. This is why the results marked with a * in Table 2 can not be counted as best known. Considering soft time windows could be switched off making time windows shorter and easier to prune. Having all trucks available in one depot and having identical trucks simplifies the decision on which truck to take. However some pruning is then not possible like not assigning orders to trucks that are too heavy or too big.

Given this, the comparison to benchmarks can only be an indicator. This is why we only added one instance for each of the six classes available in [Lim, 2010]. Anyhow, the main focus here is on optimizing real world data. As can be seen, the optimization approach is highly sensitive to the amount of orders on a single route as has been stated in Section 3.2. The corresponding amount of orders per truck is 3.1 in the optimized real world scenario.

4.2 Characterization of the real world data set

The real world problem data set consists of overall 1736 vehicle definitions starting at 248 different locations, and 2137 orders with pickup and delivery locations in six different countries across Europe. The orders define overall 921 different locations. Since this statistic is not taking the corresponding time windows into account, we decided to provide a more accurate statistic by combining a location with its corresponding date, to a so called location-date. The combination of all pickup locations with their earliest pickup date and respectively all delivery locations with their earliest delivery date results in 2113 different location-dates. Real world data also includes missing and/or implausible values, which have to be handled by the system. For this reason, carrier specific processing rules are used to clarify such situations most likely to what a human dispatcher would do.

To get a general impression of the specific data set, the average and standard deviation to all significant attributes are listed in Table 3. While the pickup time windows are often defined more precise, most delivery time windows are either missing one limit in data, or sometimes both, which leads to a high average (towards the default value of one week) with a relatively small standard deviation compared to the pickup time windows. It also seems hard to find a good indicator for reasonable pruning of the search space. According to the average and standard deviation of the capacity demand of the orders (loading meters and weight), in relation to the capacity of the available vehicles, around 95% of the orders could be served by any arbitrary vehicle. In fact 253 orders (11.8%) have a bigger capacity demand than the smallest available vehicle. But since the smallest vehicles contribute by just 0.12% to the total vehicle fleet, this theoretical potential gets again negligible. The orders cover a 20 day period (from the earliest earliest-pickup to the latest latest-delivery).

Instance	vehicles			distance			orders	avg. orders per	time (s)
	own	best	off	own	best	off		vehicle (best)	
lc1101	100	100	0%	42 460	42 488.66	-0.1%*	527	5.27	65
lr1101	95	100	-5%*	70 242	56 903.88	+19%	527	5.27	95
lrc1101	104	84	+24%	62 887	49 315.30	+27%	527	6.27	170
lc2101	39	30	+30%	34 282	16 879.24	+103%	507	16.8	589
lr2101	30	19	+58%	89 454	45 422.58	+97%	503	26.5	1 021
lrc2101	43	22	+95%	66 943	35 073.70	+91%	507	23.0	717

Table 2: Application to benchmark instances

Orders	avg	stddev
Distance (km)	666.52	334.45
Loading meters	5.65	5.45
Weight (kg)	7 428.26	8 196.43
Pickup time window (h)	102.30	81.70
Delivery time window (h)	155.35	43.46
Service time (min)	90.00	0.00
Vehicles		
Loading meters	14.93	0.26
Weight (kg)	26 850.81	537.67

Table 3: Analysis of the real world data set

4.3 Optimization results

The availability of the human dispatcher's delivery plan allows a validation with respect to real world scenario. Table 4 shows the result statistics for insertion heuristic and tour improvement heuristic with respect to the human delivery plan. While the insertion heuristic itself is just applicable on start up to build an initial solution, the tour improvement heuristic can be applied upon both plans. As mentioned before, the objective function during optimisation was the total cost (transportation and constraint costs). In a real world scenario, apart from the total cost, other parameters like the average utilisation of the vehicles, the overall driving km, the empty km and the number of time window violations are additionally used to measure the quality of a delivery plan. In Table 4 the average utilisation of a vehicle was calculated as capacity utilisation per driven km. All results were computed single threaded on an ordinary PC (Intel Core 2 Duo @ 2.8 GHz). Runtime measurements correspond to this hardware.

The insertion heuristics was able to save more than 25% of the costs compared to the plan created by human dispatchers. In more detail, since the transportation costs are directly related to the load of a vehicle together with the driven distance, most of the cost savings have to be reached by a higher utilisation. In this case, the insertion heuristic was able to rise the average utilisation from initially 45.1% up to 76.2%. In terms of costs of the matrix cost model we used, this results in a cost difference around 25% to 30% - depending on the driving distance. This and the reduction of overall kilometres are the most significant cost reduction factors. The main reason why human dispatchers fail to achieve the same result quality is most likely that not all orders are visible to them. Typically dispatchers are organised in regional business centres and have limited insight into orders of other regions to keep the assignment problem tractable to humans.

The relatively low number of violations in the human plan was reduced by another 50%. The reduction in number of used vehicles is due to the higher utilisation as well as using the same vehicles more often. A runtime of 18 minutes is definitely acceptable to logistic companies.

The tour improvement heuristics was applied to the human delivery plan, in order to see its potential upon a hand made solution. As shown in Table 4 it was able to improve in all of the previously listed comparison criteria.

The best results are achieved by running tour improvement heuristics on the solution created with insertion heuristics saving 26.9% of the costs. Three hours of runtime could be problematic in dynamic situations, but the heuristics is incremental and can be interrupted at any time.

5 Conclusions and Future Work

We were able to show, that our heuristics are able to deal with the instance size and complexity of a real world m-PDPSTW. The reported results indicate that the insertion heuristic is efficient in building initial solutions compared to the delivery plan created by human dispatchers. It can be observed that the sequential application of both heuristics significantly reduce the overall transportation cost.

It is our strong believe that it is of big value to compare optimization results with human performance. It boils down to the question if optimization should address theoretically uninteresting but practically important issues like drive time regulations or real street distances. Only by doing so, we will practically benefit from work done in this area. In the absence of exact methods to solve such instance sizes, the first ones able to solve these problem instances to compare with are human dispatchers.

The results presented in this paper are non-dynamic, i.e. they do not take the time into account at which the data was available to the system (see [Azi *et al.*, 2010]). The optimization algorithms used, however, can easily be adjusted to deal with dynamic versions of the problem. The time calculation has to be adjusted to take the current time into account. Nodes already in the past are skipped during calculation. Orders have to be inserted by the date they are known to the system. The insertion heuristics works unchanged and produces 985,756 cost on the data presented in Section 4 which is 1.8% off the non-dynamic result and still more than 20%

	Human plan	Insertion heuristic		Impr. heuristic on Human		Impr. heuristic on Insertion	
Transportation Cost	1 300 233	968 658	(-25.5%)	1 030 162	(-20.8%)	950 916	(-26.9%)
Driving km	1 246 771	885 063	(-29.0%)	951 390	(-23.7%)	873 027	(-30.0%)
Empty km	26 338	9 435	(-64.2%)	5 545	(-79.0%)	8 579	(-67.4%)
Utilisation (%)	45.1	76.2		61.2		76.1	
Violations	47	23	(-51.1%)	7	(-85.1%)	19	(-60.0%)
Vehicles	1 736	699	(-59.7%)	1 111	(-36.0%)	697	(-59.9%)
Transported Orders	2 137	2 137		2 137		2 137	
Runtime(min)	-	18		180		180	

Table 4: Comparison of human and optimised delivery plans

better than the human dispatchers. We are currently working on changing the high level workflows to include tour improvement heuristics. The main changes affect when tour improvement heuristics are triggered and what routes to select.

Currently we assume that every order is transported only on one route serviced by only one vehicle. In reality, an order on a certain vehicle might be exchanged with another vehicle while in transit. The mode of transport could also differ. In such cases, for example, it would be cheaper to transport large number of orders by rail than individually transporting them by road. The orders in the example would have to be checked for partial transport on adjacent routes. We refer to the problem as "inter-modal heterogeneous m-PDPSTW". It can be observed that the already large combinatorial space enlarges further. The analysis of performance and behaviour of known methods would allow us to understand the problem space better and improve our techniques.

References

[Azi *et al.*, 2010] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. A dynamic vehicle routing problem with multiple delivery routes. *CIRRELT-2010-44*, (44), 2010.

[Chandra *et al.*, 1994] Barun Chandra, Howard karloff, and Craig Tovey. New results on the old k-opt algorithm for the tsp. *5th ACM-SIAM Symposium on Discrete Algorithms*, pages 150–159, 1994.

[Dorer and Calisti, 2005] K. Dorer and M. Calisti. An adaptive solution to dynamic transport optimization. In Michael Pechoucek, Donald Steiner, and Simon Thompson, editors, *AAMAS 2005 proceedings*, Utrecht, 2005.

[Dumas *et al.*, 1991] Y. Dumas, J. Desrosiers, and F. Soumnis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

[Helsgaun, 2006] Keld Helsgaun. An effective implementation of k-opt moves for the lin-kernighan tsp heuristic. *Writings on Computer Science*, (109), 2006.

[Jaw *et al.*, 1983] J. Jaw, A. Odoni, H. Psaraftis, and N. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research B*, 20B(3):243–257, 1983.

[Laporte and Osman, 1995] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[Lawler *et al.*, 1985] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York, 1985.

[Lim, 2010] Li & Lim. Li & lim benchmark. Website, 2010. `http://www.sintef.no/Projectweb/TOP/Problems/PDPTW/Li--Lim-benchmark/`.

[Nanry and Barnes, 2000] William P. Nanry and J. Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research*, Part B 34:107–121, 2000.

[Papadimitriou and Steiglitz, 1982] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[Parragh *et al.*, 2008a] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems: Part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[Parragh *et al.*, 2008b] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[Psaraftis, 1995] H. Psaraftis. Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, 61:143–164, 1995.

[Ropke *et al.*, 2007] Stefan Ropke, Jean-Francois Cordeau, and Gilbert Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007.

[Savelsbergh and Solomon, 1998] MWP. Savelsbergh and M. Solomon. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

[Solomon, 2005] M. Solomon. Vrptw benchmark problems. Website, 2005. `http://w.cba.neu.edu/~msolomon/problems.htm`.

Workflow Resource Allocation through Auctions

Albert Pla[1], **Beatriz López**[1] and **Javier Murillo**[2]
[1]University of Girona, Girona, Spain
{albert.pla,beatriz.lopez}@udg.edu
[2]Newronia, Girona, Spain
javier.murillo@newronia.com

Abstract

Nowadays business processes of the manufacturing industries are becoming more complex delocalizing its production plants and outsourcing more parts of their production processes. This involves a lot of uncertainty, making the production planning, control and resource allocation complex. To deal with that issue, we propose a methodology for allocating resources into distributed manufacturing environments using a multi agent workflow management system and reverse sealed bid auctions. In this paper two different auctioning strategies are presented, one for reducing economic costs and one for reducing production time. In order to test our approach we simulated different situations using different kinds of workflows and resources, getting promising results both in economic and time terms.

1 Introduction

The economy globalization is driving many manufacturing industries towards the decentralization of their production processes. This means not only to distribute the production into different factories and production plants but also to outsource some steps of the chain production, increasing the complexity of the supervision and the planification of the production processes.

The production chain is no longer able to be controlled by a single entity. The status of the production resources (e.g. technicians, transports, services, etc.) is unknown as they can be managed by different departments inside the organization or even by outsourcing companies which are in charge of dealing with a certain part of the business process. Moreover, the intervention of third party elements also difficults the cost optimization creating a confrontation between the manufacturers, which try to obtain the lowest price and the higher quality in the market, and the outsourcing companies which tries to maximize its benefits and its occupation. Each outsourcing company has it's own schedule with customers which cannot be seen by others due to privacy issues. An example of this situation is a medical device maintenance service of an hospital. In a medical environment, medical devices need different maintenance operations such as revisions, reparations, reconfigurations, etc. Some of them can be scheduled in advance, nevertheless others such as fault reparations or contingencies cannot be planned thus affecting the normal development of the hospital. This tasks must be carried out by qualified technicians which can be part of the hospital staff itself but some times outsourcing technicians are required.

Production methodologies such as *Lean Manufacturing* [Shah and Ward, 2003], which are strongly customer-oriented, postulate that production must strictly satisfy customer demand and specifications to avoid creating any unnecessary values and without resorting to unnecessary work. This approach encourages the interaction between the manufacturer and the customer and it also empowers the customization of the final product. Thus, the flexibility in the production chain allows the producer to introduce modifications into the original design without affecting other tasks. The Lean philosophy can be used in embedding processes where the customer can personalize its order, assembling only the required pieces and without having pre-assembled stock, saving storage space and reducing the number of waste stocks. To meet these requirements, production can be realized under demand: allocating resources on real time without taking into account possible future processes which could never be started. From the scheduling point of view, this means that the planing for a business process is not planned until it is demanded and that the business process do not allocate a resource until the task which requires the resource is about to start.

The main characteristics of these new manufacturing scenarios are dynamism, decentralization, collaboration with outsourcing third partys, contingency robustness and customer orientation. Therefore a lot of uncertainty is involved, making the production planing and control complex. Our work concerns the research for new tools that support managers in these environment. We propose a methodology for allocating resources into distributed manufacturing environments based on compounding workflows with multi agent system (MAS) and auctions for resource allocation being resource used from a wide scope: a technician intern to a company, a service provided by third party company, etc.

On the one hand, MAS provide an infrastructure in which different companies can be coordinated to deploy an activity while maintaining their schedule and customers appointments in privacy. On the other hand, auctions offer companies the

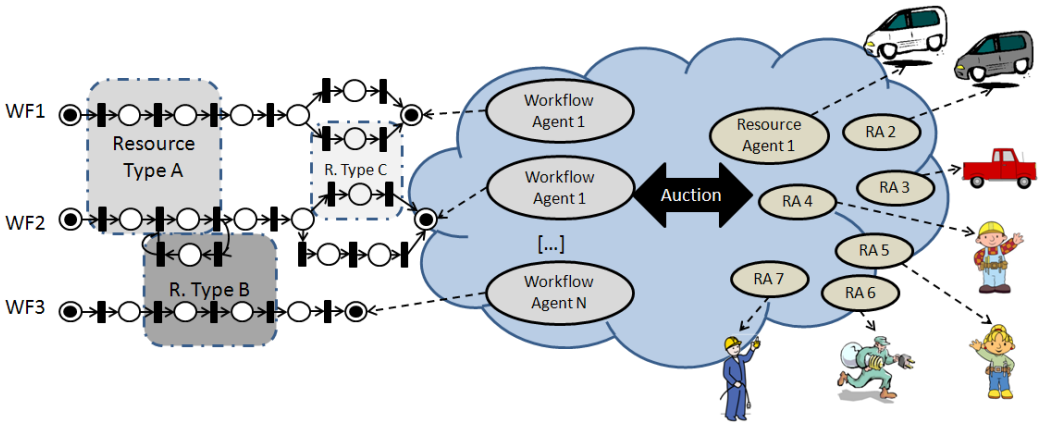


Figure 1: Schema of the system architecture: each workflow type is monitored by a Workflow Agent while each resource is represented by a Resource Agent.

chance to compete for providing a resource or service to a manufacturer without revealing private information [Chevaleyre *et al.*, 2006]. Thus, workflows are handled by agents which, in turn, use auctions for negotiating for the resources they need to accomplish their activities.

The use of auctions for resource allocation benefits the manufacturer not only decreasing the outsourcing prices, but also increasing the occupation of its own resources. On the one hand, workflows will be able to minimize the cost of the resource by comparing the different bids offered by the available resources and they will be also able to decide which resources are more suitable to fit in their timing compromises without the need of accessing to their agendas. On the other hand, when internal resources receive incentives for they use, auctions balance its workload as resources tend to maximize its occupation.

2 Background

To manage the evolution and the interactions of the business processes it is important to accurately model the steps to follow in the activity, the resources needed and the flow of information between the different parts involved (suppliers, manufacturers, clients, etc.). Workflows provide a way of describing the order of execution and the dependent relationships between the constituting activities of the business processes [Tick, 2002]. Workflows usually model single and unique business processes, nevertheless, in real life environments, workflows are rarely executed individually. Workflows usually run concurrently, sharing a limited number of resources which some times are provided by third party companies.

A workflow consists in a graph of interconnected actions which represents the tasks and interactions to be realized by a mechanism, a person, a staff, an organization, etc. Workflows can model business process, exchange of messages and software procedures or information.

A workflow instance is a workflow which is being executed in a concrete time instant. For example, a workflow can model the business process required to do maintenance in a medical equipment; then, when a medical device requires a maintenance operation, a workflow instance is created. When several workflows are coexisting in a common framework (e.g. an organization, an industry, a server, etc.) where they share resources, actors or information they are called a workflow environment. Workflows can be controlled and monitored by a workflow management system (WMS).

WMS manages and monitors the different tasks which take place inside an organization or a workflow environment. It is responsible for monitoring the status of the different workflows and to store in a log the different events related to the workflow deployment. WMS can also be responsible of the assignment of resources to workflows and their schedules.

3 Related Work

There are several previous works related to the application of agents to workflow management systems to support cooperation activities. For example, [Juan *et al.*, 2009] uses agents to facilitate the collaboration of multi-disciplinary workgroups in a company for design products following a concurrent new product development strategy They start with predefined requirements of cooperation, and the multi-agent approach is used to achieve such requirements. Conflicts can arise when there are different points of views on cooperative tasks, and they are solved by specific cooperation diagrams. Our proposal includes different ways of conflict resolution, by means of auctions. In [Guo *et al.*, 2008] a decentralized multi-agent architecture is proposed workflows. Particularly, the authors face the problem of interoperability between heterogeneous agents and they propose the use of business modeling languages as BPEL4WS [Wohed *et al.*, 2003]. In our case, we are assuming that agents are able to understand each other; so this assumption can be leveraged with the complementary approach offered in [Guo *et al.*, 2008].

A founder work in this field is [Jennings *et al.*, 2000], in which the authors explore the use of agents to enact cooperation at the business level thanks to the advent of Internet. In this case, workflows are not fixed, but agents represents collections of services that can be combined (by agents negotiation) to compound business process. This approach is a

radical one, since it requires from a novel point of view of current business. Our approach is a step forward to achieve such a revolution, but by following an evolutionary approach. So we are starting from the current tools that are in the industry (workflows) and provide a way to make them more flexible thanks to agents. This less-drastically approach could be more useful to non-Internet based business models.

Other approaches regarding workflows and agents can be found in the recent published survey [Delias *et al.*, 2011]. Our approach differs from the previous ones in the way resource conflicts are being handled, taking into account penalties. Moreover, we respect current WMS while proposing an extension that takes advantage of current advances on agents, which improves monitoring capabilities of WMS (reducing, for example, delays).

4 Auction-based workflow resource allocation

We propose to handle both workflows and resources within a multiagent WMS which monitors and organizes the course of the manufacturing process. For that purpose two different kind of agents are used: the workflow agents (WA), one per workflow, and the resource agents (RA), one per existing resource. As Figure 1 shows, WA supervises the workflow it represents and when a resource is needed it summons a reverse sealed bid auction [Amelinckx *et al.*, 2008] indicating the desired conditions (minimum starting time, maximum ending time, resource type, etc.) and a penalty to pay if a contract is broken, becoming, thus, an auctioneer. Then, resource agents evaluate their agendas and decide to participate or not to the auction depending on the auction conditions and the possibilities the resource have to finish the task on time.

Auctions have three main elements:

- **Winning Determination Problem** (WDP), handled by the auctioneer (WA).
- **Bidding Policies**, which are the bidding strategies followed by the bidders (RA).
- **Pricing Mechanism**, which defines the payment methods between agents.

In our approach we follow a reverse sealed bid auction, meaning that the role between the auctioneers and the bidders is reversed as the auctioneer is the one who wants to buy a good, not to sell it. While ordinary auctions provide suppliers a chance to find the best buyers, reverse auctions give buyers an opportunity to find the lowest-price supplier and improving the chances of improving a fair market value. To encourage the resources to bid according to the market value, bidders submit its proposals simultaneously, without knowing other participants bids and the one with a lowest value becomes the winner. To prevent fraudulent bids, agreement repudiation or delays into the ending time agreement, auctions are endowed with a penalty mechanism that allows the agent to fine the resources which do not accomplish the established agreements.

4.1 Workflow Agents: The Auctioneers

Workflow agents are the ones responsible for monitoring running workflows and for obtaining the necessary resources to finish them in the required time. Each WA monitors just one workflow, which is modeled using high level Petri nets (PN). Each time a workflow should be started, a token is add to the start node of the PN. For example, a workflow can model the repair process of a medical device. Thus, each time a medical device needs to be repaired, a token is added to the workflow, meaning the instantiation of the workflow.

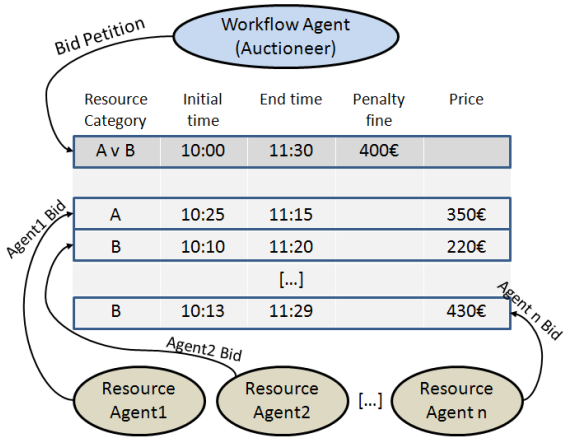


Figure 2: Example of a resource action. If the WDP considers the lowest price as the winning policy, Resource Agent 2 would be the winner, otherwise, if time is considered, Resource Agent 1 would be the winning resource

Workflow modeling using high level Petri nets has been broadly studied [Alt *et al.*, 2006]. As our work is specially focused on resources, we need to take care of the kind and number of resources needed for every task inside the workflow. In order to satisfy this requirement we extended the Petri Net representation with a new *resource* element [Pla *et al.*, 2011]. We called this extension *resource-aware Petri nets* (RAPN). RAPN incorporate resources to high level Petri nets. Resources are related with sets of consecutive transitions where the first transition is the one which allocates the resource and the last is the one which releases it. If there are not available resources of the required type by a transition this transition cannot be fired until a resource of the desired type can be used.

When a resource is needed in a workflow activity, the WA must obtain a suitable resource to satisfy the agent requirements. By suitability we mean that activities have not a specific resource assigned to them (e.g. technician 1) but features of the resource (e.g. a technician with a specific license). In the call, the auctioneer specifies the different attributes to be fulfilled by the resource willing to deploy the task (e.g. minimum resource license, starting time and ending time.). According to the interests of the workflow agent the determination of the auction winner can be solved in two different ways:

- **Balanced Strategy:** The winner of the auction is the bid which offers the lowest price (in Figure 2 winner agent would be Resource Agent2). This strategy decreases the costs for the workflow agent while promotes a balanced market price as bidders tend to offer reasonable prices in

order to avoid the loss of customers. The balanced strategy is suitable for dealing with outsourcing resources as it can obtain fair prices. However, it is also useful when dealing with inside company resources as when an equilibrium market price is reached since bidders try to maximize its benefits by increasing its occupation arising the productivity of the local company resources.

- **Delay Prevention Strategy:** This strategy favors the shortening of the workflow timings and in a reduction of delays by setting the auction winner taking into account the ending time (in Figure 2 winner agent would be Resource Agent1). By setting as winner the bid with an earliest ending time, the ending time for the workflows will be shortened but the economic cost will be higher than the one obtained in the balanced strategy. This fact makes this strategy specially indicated for dealing with the own company resources where costs are not the hardest constraint.

As seen in Figure2, the policy adopted in the WDP will change the system behavior and benefits.

4.2 Resource Agents: The Bidders

The interests of the resources are also defended by agents. Each resource is represented by RA which is defined by its category (the set of tasks feasible by the resource), the schedule of tasks (agenda), an estimation of the time needed to achieve certain procedures and its time constraints. The main goal of each agent is to maximize its benefit and, in consequence, to maximize and to capitalize its occupation.

Resources are free to decide whether to participate in an auction or not and to set the bid they consider convenient, however the bidding strategies they choice must be in concordance with their aims. RAs must bid taking into account their agenda, the benefits that behave their schedules and the penalties that accomplishing their timetables. These facts will define the character of the RA. For example an agent could decide to cancel one of its planned activities if the benefits of accepting another task and paying the corresponding penalty overcomes the profits generated by the scheduled task. Agents can also offer risky bids, using its experience they can analyze the probability of ending a task on time; as RA are charged with penalties when they do not end tasks on time, they can assume the risk arising the bid price according to this probability in order to amend the fine in case of delay. As a first approach we have defined bidders capable to adapt their prices to the market, increasing its prices when the demand is high and decreasing them when demand is lower[Lee and Szymanski, 2005]. Fines evaluation will be included in a future work.

5 Experimentation

In this section the previously presented methodology is tested to evaluate the Multi-agent WMS performance and benefits when using different auctioning strategies. Using the workflow simulation engine presented in [Pla *et al.*, 2011] a workflow environment with common resources have been simulated. The experiments are evaluated in terms of economic

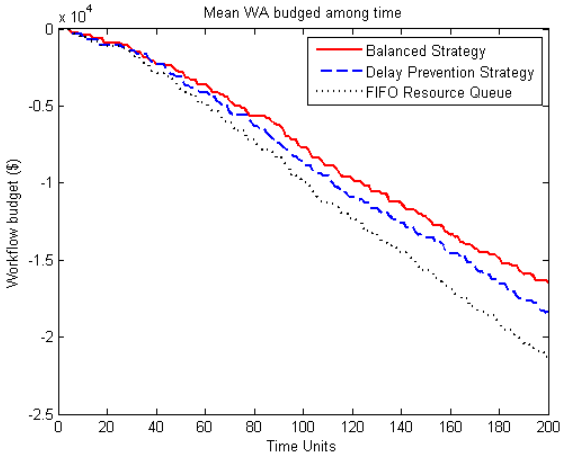


Figure 3: Mean Workflow agents' budget in Scenario1

cost (agent benefits and costs), workload balancing and delays in the process execution.

5.1 Experimental Set Up

To test the performance of our system we modeled and simulated a set of three synthetic workflows. Each of these workflows is composed by four different tasks which have a duration compressed between 10 an 15 time units and needs a resource of a randomly assigned category (between A and F). In consequence, each workflow has a duration between 40 and 60 time units and requires between 1 and 4 different resources. In the simulation, the number of tasks that will be executed is unknown as it simulates and organization where workflows are not scheduled, they arrive under demand.

The two previously defined auctioning strategies, *Balanced Strategy* and *Delay Prevention Strategy*, have been evaluated. Besides the auction resource allocation methodology, a simpler scheduling method have been used in order to compare the two methodologies. For that purpose a *First In First Out* resource queue have been used. Every time a workflow needs a resource of a concrete type, it checks a list of the system resources and uses the first available. In cases of two workflows requiring the same resource, according to the FIFO strategy, the first workflow to ask for the resource would be the first one to be served.

To evaluate our methodology two different scenarios have been tested:

Scenario1: This scenario simulates a workflow environment with four different resources during 200 time units and with a probability p of starting a new workflow $p = 0.2$. Each resource can perform tasks of three different categories (RA1: A-B-C. RA2:A-B-D. RA3:C-E-F. RA4:D-E-F). This experiments shows the behavior of the methodology in a common situation: the number of resources is lower than the number of workflows that requires them.

Scenario2: This scenario repeats the previous experiment but with a significant difference, this turn, each resource can perform any type of task (A-F), increasing competency between resources. This scenario lasts 200 time units and with

a probability p of starting a new workflow $p = 0.2$. The aim of this experiment is to evaluate the behavior of the system when there is a high competitivity between similar agents and how the workload is distributed.

Both scenarios have been repeated 20 times for each auctioning strategy and for the FIFO resource queue in order to obtain significant results.

5.2 Results

The results of the different experiments are shown in Tables 1 and 2. In them, information about economic costs and delays produced are given. As each experiment has been repeated several times, the results are expressed in terms of mean and coefficient of variation. Some graphical examples are also presented.

Table 1 presents the results of the first scenario. Regarding delays, as it was expected, it can be seen that, while the *Balanced Strategy (BS)* and the FIFO resource queue (FRQ) produce a similar number of delays (5,6 and 6,4 in average respectively), using the *Delay Prevention Strategy (DPS)* significantly reduces the number of delays (2,2). Second, from the economic point of view, Figure 3 shows how the use of the BS reduces the money spent by WAs and, as a consequence, the average earns by RAs also decreases.This figure also shows that the DPS also reduces the costs respect the FRQ. Another interesting fact which can be observed in Table 1 is that using the BS the variance of the resources benefits is up to 4 times lower than in the other two strategies, indicating that the resource agents have offered similar prices, reaching a more balanced price market.

In Table 2 we can observe how the results of the second experiment are similar to the previous one: DPS reduces the number of delays during workflow executions while the use of agents reduces the costs for the workflows. A relevant point in this experiment is the variation coefficient of the resource agents' incomes. We can see how the use of auctions (specially using a BS) decreases the resource prices and reduces the variability of these costs respect a FRQ. In this sense it is important to notice that when all the resources of an environment have the same properties, the BS tends to homogenize the cost of the resources: the coefficient of variation of resource benefits is $6,67\%$ against the $18,33\%$ of the DPS and the $33,96\%$ of the FRQ. Comparing this coefficient with the one obtained in the first scenario (where all the resources had different capabilities) by the BS $10,74\%$ we can corrobarate how the more similar the resources are the more balanced the price market becomes. Finally, Figure 4 illustrates how the workload of the agents have been also balanced conversely the FRQ. The plot shows gaps in the RA occupation and how in the FRQ resources are not starting to work until previous resources in the resource list are occupied while using BS the occupation of the resource is rather more balanced.

The performed experiments have shown how the BS decreases the costs for the WAs and how equilibrates the benefits between similar RAs. Moreover, BS can balance the workload between agents when they have similar capabilities. Regarding DPS, we observed that it significantly reduces the number of delays respect FRQ and it also reduces the economical costs, although not as much as BS.

Table 1: Results of a common situation in therms of resource availability.

	BS	DPS	FRQ
Delayed Workflows	5,60	2,20	6,40
WF1 Spent Money	15846,00	16308,00	20758,00
WF2 Spent Money	17806,20	20723,40	19814,00
WF3 Spent Money	19280,60	22146,00	22028,00
Total Spent Money	52932,80	59177,40	62600,00
Mean Spent Money	17644,26	19725,80	20866,67
Std Dev	1723,02	3044,17	1111,00
Coef. Variation (%)	9,76	15,43	5,32
Resource1 Earned $	15068,80	18005,60	16880,00
Resource2 Earned $	12228,80	21692,40	19800,00
Resource3 Earned $	11997,60	10148,20	11616,00
Resource4 Earned $	13637,60	9331,20	14304,00
Total Earned Money	52932,80	59177,40	62600,00
Mean Earned Money	13233,20	14794,35	15650,00
Std Dev	1422,26	6036,77	3503,34
Coef. Variation (%)	10,74	40,80	22,38

Table 2: Results of a scenario where all the agents are competing to provide the same resource typology.

	BS	DPS	FRQ
Delayed Workflows	5,80	2,60	8,20
WF1 Spent Money	16003,40	18229,00	23907,40
WF2 Spent Money	19602,60	20695,80	24046,80
WF3 Spent Money	22417,20	23751,00	25802,60
Total Spent Money	58023,20	62675,80	73756,80
Mean Spent Money	19341,07	20891,93	24585,60
Std Dev	3214,89	2766,22	1056,26
Coef. Variation (%)	16,62	13,24	4,29
Resource1 Earned $	13238,00	13897,80	23240,20
Resource2 Earned $	14859,20	18832,20	24408,00
Resource3 Earned $	14668,40	12665,40	12354,60
Resource4 Earned $	15257,60	17280,40	13754,00
Total Earned Money	58023,20	62675,80	73756,80
Mean Earned Money	14505,80	15668,95	18439,20
Std Dev	880,12	2872,93	6262,31
Coef. Variation (%)	6,67	18,33	33,96

6 Conclusions and Further work

This paper concerns an important problem for the manufacturing industry: how to allocate internal and foreign resources or services in a decentralized production processes while maintaining customers appointments and providers schedules in privacy. Our work is also related with flexibility issues in the production chain as provides mechanisms for managing customer-oriented production methodologies such as *Lean Manufacturing*. We proposed a methodology which combines workflows with MAS and auctions. On one side MAS offers a framework where companies can coordinate and deploy their activities without jeopardizing privacy or flexibility. On the other side, auctions can be useful for resource allocating in different senses: minimizing costs of external

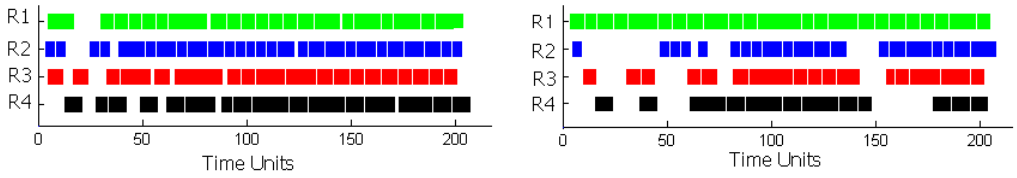


Figure 4: Resource occupation on Scenario2. Left: Occupation using BS. Right: Occupation using FRQ

services, reducing timings and balancing the occupation of internal resources.

Our approach presents a workflow management system which handles both workflows and resources using two kind of agents: workflow agents and resource agents. Resource allocation is carried out using auctions: each time a WA requires a resource, it calls a reverse sealed auction with its desired conditions while RA compete to win the auction. Two different strategies for the auctioneer agent have been presented: Balanced Strategy and Delay Prevention Strategy. The first one is focused on the resource costs and it is suitable for decreasing the resource prices (when dealing with outsourcing resource) and to equilibrate the workload balance (when allocating internal resources). The second one is centered on the time needed to finish a task and its purpose is to reduce the number of delayed workflows during the execution of several concurrent business processes.

To test the performance of our approach we simulated different synthetic workflows with different resource availability conditions. The workflows have been managed using a multiagent workflow management system endowed with the two presented auctioning strategies. The results show how the two presented auctioning strategies result in economic cost reduction, balanced market price, workload balancing and delay reduction, showing that BS is indicated for improving the three first mentioned measures while DPS enhances the last one. These results encourage to develop a new strategy which balances the 4 measures, finding a compromise between BS and DPS.

As a future work we plan to include fine evaluation to bidder agents. Moreover we want to elaborate new auctioning strategies and to consider including multi-attributive auctions. Another step would be to improve certain MAS capabilities of the system such as trust or reliability on the resources based on historical information, which will help us dealing with cheating bidder agents.

Acknowledgements

This research project has been partially funded through the project labeled TIN2008-04547 and BR10/18 Scholarship of the University of Girona granted to Albert Pla.

References

[Alt *et al.*, 2006] M. Alt, S. Gorlatch, A. Hoheisel, and H.W. Pohl. Using high-level petri nets for hierarchical grid workflows. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 13, Washington, DC, USA, 2006. IEEE Computer Society.

[Amelinckx *et al.*, 2008] I. Amelinckx, S. Muylle, and A. Lievens. Extending electronic sourcing theory: An exploratory study of electronic reverse auction outcomes. *E.C.R.A.*, 7:119–133, 2008.

[Chevaleyre *et al.*, 2006] Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lematre, N. Maudet, J. Padget, S. Phelps, J.A. Rodrguez-aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:2006, 2006.

[Delias *et al.*, 2011] P. Delias, A. Doulamis, and N. Matsatsinis. What agents can do in workflow management systems. *A.I.R.*, 35:155–189, 2011.

[Guo *et al.*, 2008] L. Guo, D. Robertson, and Y. Chen-Burger. Using multi-agent platform for pure decentralised business workflows. *Web Intelli. and Agent Sys.*, 6:295–311, August 2008.

[Jennings *et al.*, 2000] N. R. Jennings, T. J. Norman, P. Faratin, and B. Odgers. Autonomous agents for business process management. *Journal of Applied Artificial Intelligence*, 14:145–189, 2000.

[Juan *et al.*, 2009] Y.C. Juan, C. Ou-Yang, and J.S. Lin. A process-oriented multi-agent system development approach to support the cooperation-activities of concurrent new product development. *Comput. Ind. Eng.*, 57:1363–1376, November 2009.

[Lee and Szymanski, 2005] J. Lee and B.K. Szymanski. A novel auction mechanism for selling time-sensitive e-services. In *IEEE Conference on ECommerce Technology (CEC'05*, pages 75–82. Press, 2005.

[Pla *et al.*, 2011] A. Pla, B. Lopez, J. Melendez, and P. Gay. Petri net based agents for coordinating resources in a workflow management system. In *ICAART*, pages 514–523, Rome, Italy, February 2011.

[Shah and Ward, 2003] R. Shah and P.T. Ward. Lean manufacturing: context, practice bundles, and performance. *op. Management*, 21(2):129–149, 2003.

[Tick, 2002] J. Tick. Workflow model representation concepts. *Int. Symposium of Hungarian Researchers on Comp. Intelligence WF Model Representation Concepts*, 7, 2002.

[Wohed *et al.*, 2003] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of web services composition languages: The case of "'bpel4ws"'. In *ER03, LNCS 2813*, pages 200–215, 2003.

Stochastic programming as a tool for emergency logistics in natural floods

Patricio Lamas, Rodrigo A. Garrido
Universidad Adolfo Ibanez
Santiago, Chile
rodrigo.garrido@uai.cl

Abstract

Catastrophic events pose hard logistics chalenges, because the transportation and/or communication networks are damaged and supply chain capacities are affected. Under these circumstances, the demand patterns change in shape and magnitude. Thus the standard array of consumption typically forecasted by producers and distributors is no longer valid and hence standard logistics practices are unable to deliver goods and services on time at the right place. This article presents a modeling framework to assist decision makers in the planning stage of immediate assistance of natural disasters victims. The modeling framework gives an optimal inventory policy for emergency supplies and optimal fleet distribution. In this article, floods are the only possible events in the foreseeable future. It is possible to establish a stochastic process to represent the probabilistic occurrence of floods in different zones throughout a year.
The mathematical model that optimizes the inventory levels flows and vehicles allocation is a large size stochastic integer programming model. The model is (approximately) solved through sample average approximation. An example is provided.

1 Introduction

Most of the logistics systems are designed to operate under standard conditions, i.e., when the transportation and communication networks are fully operative, the suppliers are able to deliver what they are asked and the demand patterns fluctuate within (somewhat) known bounds, as well as availability of human resources and vehicles to distribute products and services from production sites to consumption points. Even under this scenario, the logistics strategy and operation are rather complex tasks both for the size of the instances to be solved and for the type of models to be solved when trying to optimize sensible variables such as inventory levels or sequences of vehicles stops under time constraints. However, this already complex situation becomes much more cumbersome when there is uncertainty on some (or all of the) system´s components. That is the case in the logistics of emergencies right after a natural disaster has occurred. Typically, there will be victims that need prompt attention in several dimensions: health care, food, water, safety, and childcare among others. However, the means to deliver the assistance may have been severely damaged by the natural disaster and hence the standard supply plans of the various industries involved in the provision of these goods do not hold in the disastrous scenario. Emergency logistics plans consider pre-positioning supplies and vehicles to reduce the travel times as well as transportation capacity; in addition, an optimal assignment of supplies and vehicles to a given location assures the minimization of total systems cost while at the same time satisfying time constraints.

Natural disasters fall under the category of low-probability high-consequence events, both in terms of human, animal and economic losses. In this article we focus only on disasters that exhibit a seasonal and/or spatial recognizable pattern that can be adequately represented by a (computationally) tractable stochastic process. That is the case of natural floods, which occur mostly during the rain seasons and affect mostly low lands or plain enclosed areas. Thus, their occurrence pattern can be represented by a certain space-time probability distribution model. Floods can be generated by a number of factors, where excessive rain ranks among the most common natural causes of floods in Chile, affecting mostly cattle and rural towns.

1.1 Objective

The aim of this article is to develop a modeling structure to assist the decision makers in tactical aspects of the emergency logistics after a flood occurrence, answering the following questions: What type and quantity of products should be kept in stock at each location and period within the area of interest and time horizon respectively? and, How to transport the supplies from the stocking facilities to the demand points at minimum cost and time?

1.2 Literature Review

In spite of the enormous relevance that emergency logistics has for the current society, the number of publications in this topic is considerably lower than that of commercial logistics. The most cited publication on emergency logistics, related to natural disasters, is [Ozdamar *et al.*, 2004], which

focuses on operational decisions in natural disasters in general. [Fiedrich *et al.*, 2000], presents a study for strategic decisions in the case of earthquakes. [Chang *et al.*, 2007], develops a strategic model for the case of floods, considering different scenarios. The type of model developed in this article falls into the category of stochastic programming. In particular, we developed a probabilistic programming model. Recently, [Pagnoncelli *et al.*, 2009; Luedtke and Ahmed, 2008], present methods to solve real-case problems involving probabilistic programming modeling.

To the knowledge of the authors, at the time this article was written, no articles have been published with developments in probabilistic programming models applied to tactical decisions in emergency logistics.

2 Problem Description

2.1 Some Characteristics of a Natural Disaster

After the occurrence of a natural disaster, a number of negative impacts reach the population in the affected area. In the case of rural areas, where cattle raising is the main economic activity, the floods endangers the animal production as well as the crops used to feed the cattle, generating a deficit in supplies. Therefore, after a flood the cattle producers will demand forage, fresh water, nutritional supplements and medicines in larger amounts than they would normally do. The latter is an emergency demand, which must be satisfied as quickly as the transportation and communication networks allow. In this case the local authority is the the responsible for delivering the emergency supplies accurately and on time. These situations typically involve a minimum scale time of 4 seasons, a spatial span between 5 and 15 zones, at least 2 types of vehicle (e.g. dry and reefer) and at least 2 types of supplies (e.g. water and solid foods). From the tactical perspective, the emergency logistics should aim to maintain levels of inventory of (a few) urgently needed goods, sufficiently high to satisfy the expected demand for them after an event, but at the same time not too high to avoid costly excessive stock. That means that the inventory levels should guarantee the satisfaction of demand to a certain confidence level, i.e., to an acceptable degree of failure. For instance, a desired objective could be to find the levels of inventory that assures that demand will be satisfied at least in the 95% of the cases. Additionally, the demand satisfaction must not only be within the desired confidence level but also must be done at minimum cost, where the cost is any generalized distance that may include response time and resource consumption, among others.

The stochasticity inherent to the occurrence of floods makes the demand function to change abruptly from zero to a high level of consumption. This is a characteristic rarely seen in the consumption of standard goods (or goods under standard conditions) and hence the standard levels and location of stocks may not be able to fulfill the demand within acceptable time limits after a flood has stroke an area.

2.2 System Agents

Products Suppliers

These agents are willing to offer a set of products of potential demand. Finished products are stocked in different locations spread out within an area significantly larger than any affected zone. If an emergency occurs, some of the products would be retrieved from one or more depots in the amounts that a minimum cost flow program recommends. Note that prices may vary between producers as well as within the same producer across time periods.

Transportation Providers

Carriers offer the service of moving goods from the suppliers depots to the demand points. They operate with a fleet of vehicles that are either parked in different locations or moving between a large number of origins and destinations. Thus, within their whole fleet, it is possible to ascribe an available fleet per location during any given period. The transportation tariffs are charged per unit of freight per travelled distance. The tariffs may vary among different carriers as well as within the same carrier across time periods.

Vehicles within a fleet are not necessarily homogeneous. Certain products need a special type of vehicle (e.g. vaccines) and hence there is a compatibility matrix which shows acceptable vehicle-product pairs.

Demand Points

They correspond to geographic zones affected by the flood or specially designated loading/unloading facilities. These zones represent the demand for products (consumption) even though the actual consumption might take place in a different location. The magnitude of the demand at these zones is formed by aggregation of individual demands for all the affected cattle producers assigned to that particular zone.

3 Modeling the Problem

Both the time horizon and the geographic scope are defined a priori by the decision maker. The geographic area is subdivided into regions and the time horizon is divided into discrete periods. The demand for products that would be generated by a flood must be forecasted on this spatio-temporal grid. Those demand forecasts, along with some initial conditions, define an instance of the problem to be solved.

At any of the regions of the specified geographic partition, at a certain time period a flood may occur with a given probability. This probability is drawn from the history of flood records for similar seasons in the past.

For any given region and time period, the probability of flood occurrence can be estimated via a Bayesian Processor of Forecasts [Kelly and Krzysztofowicz, 1994] which maps both an a priori description of the uncertainty about flood occurrence and its magnitude, and a posterior description

of uncertainty about flood occurrence and its magnitude, conditioned on a flood magnitude forecast. Note however, that the accuracy of these forecasts will largely depend on the type of phenomenon producing the flood. For instance, in forecasting hurricane-induced floods or when a severe thunderstorm is developing, there may be little uncertainty about the large amount of rainfall to be produceed, but there may be large uncertainty about its spatio-temporal location. This situation would imply significant uncertainty regarding the occurrence of rainfall over critical points in the study area within the reach of the storm.

Once the event strikes a populated area, there is an immediate generation of demand for certain basic products (e.g. fresh water and food for cattle). Then, the magnitude and composition of the demand will change probabilistically according to the initial conditions and the severity and scope of the emergency. While some of the initial conditions can be accurately known (such as number of cattle producers, inventory at hand, and operational characteristics of the potentially affected area), other conditions (such as the severity and duration of the event) cannot be predicted with acceptable degrees of accuracy. Therefore, it is necessary to fit a set of probability distribution functions to represent the magnitude of the demand and the possible scenarios that a particular realization of the event would generate.

On top of the uncertainty posed by the demand function, there is another source of uncertainty that must be taken into consideration: the capacity loss on the transportation network. Indeed, the networks normal conditions are affected not only by direct action of the catastrophic event but also for the impulsive behavior of users who may overcharge some of the arcs creating extra congestion with unnecessary trips that preclude the timely access of emergency teams. Additionally, the media exposure generates compulsive donations of unwanted items that may saturate the logistic system with the convergence of *materiel* that is both unnecessary and hard to handle.

3.1 Mathematical Formulation of the Main Problem

This section presents the formal modeling of the problem at hand, based on the conceptual description provided in the previous section.

General Definitions

The stochasticity of the demand vector imposes the definition of various probabilistic parameters:
p_i^t: Probability that a flood occurs at location i during period t.
P_i^t: Bernoulli random variable which takes value 1 with probability p_i^t.
$\rho p_{i,j}^{t_a,t_b}$: Correlation between $p_i^{t_a}$ and $p_j^{t_b}$, for locations i and j, and periods t_a and t_b.

The demand function for product p, at location i, during period t is defined as follows:

$$D_{ip}^t = \begin{cases} d_{ip}^t & \text{with probability } p_i^t \\ 0 & \text{with probbility } 1 - p_i^t \end{cases} \quad (1)$$

$$d_{ip}^t \sim F_{ip}^t(x) \quad (2)$$

$F_{ip}^t(x)$ is some probability distribution function over a real non-negative domain.

Analogous to the case of a flood, the demand for products may also exhibit significant correlation both in time and space. Accordingly, we define:
$\rho d_{i,j}^{t_a,t_b}$: Correlation between $d_{ip}^{t_a}$ and $d_{jq}^{t_b}$, for locations i and j, and periods t_a and t_b.

The Optimization Problem

The solution to this problem is aimed to assist the decision maker in tactical aspects of the emergency logistics after the flood occurrence. Thus, the model will answer the following questions:

1. What type and quantity of products (potential demand) should be kept in stock at each location and period within the area of interest and time horizon respectively?
2. How to transport the demanded products from the stocking facilities to the demand points at minimum (generalized) cost?

To answer these questions we need to establish the models parameters and variables:
I: Number of locations within the area of interest.
Φ: Set of all the potentially affected locations $i = \{1 \dots I\}$.
P: Number of products.
Π: Set of all the potentially demanded products $p = \{1 \dots P\}$.
T: Number of periods within the planning horizon.
Ψ: Set of all the periods $t = \{1 \dots T\}$.
C: Number of vehicle classes.
Ω: Set of vehicle classes $c = \{1 \dots C\}$.
D_{ip}^t: Demand for product (random variable) p, in location i, during period t.
d_{ip}^t: A realization of D_{ip}^t p, in location i, during period t.

u_c: Capacity of vehicle class c.
w_{pc}: Compatibility matrix product-class. Typical element (p, c) takes value 1 if product p, can be transported on the vehicle class c and 0 otherwise.
V_{ic}^t: Number of vehicle classes c, available in location i, at the beginning of period t.
L_{jp}: Inventory capacity at location j, for product p.
cv_{ijkp}^t: Cost of transporting one unit of product p, to the location k, originated from a supplier in location j, transported

on a vehicle available in location i, during period t.
cl^t_{ijc}: Cost of relocating a vehicle class c, sent from location i to location j, during period t.
ci^t_{jp}: Unitary inventory cost for product p, stocked in a depot located in j, during perod t.
α: Level of confidence; at least $100(1-\alpha)\%$ of the demand must be satisfied.

3.2 Variables

The following are the modeling variables:
x^t_{ijkp}: Flow of product p, sent to location k, originated from a supplier in location j, transported by a vehicle available in location i, during period t.
y^t_{ijc}: Flow of vehicle class$c$, relocated from location $i$to location j, during period t.
I^t_{jp}: Inventory of product p, kept in depot located in j, at the beginning of period t.

With the parameters and variables previously defined, the following mixed integer programming model is presented:

$$\min \sum_{t\in\Psi}\sum_{p\in\Pi}\sum_{k\in\Phi}\sum_{j\in\Phi}\sum_{i\in\Phi} cv^t_{ijkp}x^t_{ijkp} + \sum_{c\in\Omega}\sum_{j\in\Phi}\sum_{i\in\Phi} cl^t_{ijc}y^t_{ijc} + \sum_{t\in\Psi}\sum_{p\in\Pi}\sum_{j\in\Phi} ci^t_{jp}I^t_{jp} \tag{3}$$

$$P\left\{\sum_{j\in\Phi}\sum_{i\in\Phi} x^t_{ijkp} \geq D^t_{kp} \quad \forall\, k\in\Phi,\, p\in\Pi,\, t\in\Psi\right\} \geq \alpha \tag{4}$$

$$\sum_{p\in\Pi}\sum_{k\in\Phi}\sum_{j\in\Phi} w_{pc}x^t_{ijkp} \leq u_c \sum_{j\in\Phi} y^t_{ijc} \quad \forall\, i\in\Phi,\, t\in\Psi,\, c\in\Omega \tag{5}$$

$$\sum_{j\in\Phi} y^t_{ijc} \leq V^t_{ic} \quad \forall\, i\in\Phi,\, t\in\Psi,\, c\in\Omega \tag{6}$$

$$\sum_{k\in\Phi}\sum_{i\in\Phi} x^t_{ijkp} \leq I^t_{jp} \quad \forall\, j\in\Phi,\, p\in\Pi,\, t\in\Psi \tag{7}$$

$$I^t_{jp} \leq L_{jp} \quad \forall\, j\in\Phi,\, p\in\Pi,\, t\in\Psi \tag{8}$$

$$x^t_{ijkp}, I^t_{jp} \in \mathbb{R}^+ \quad \forall\, i\in\Phi,\, j\in\Phi,\, k\in\Phi,\, p\in\Pi,\, t\in\Psi \tag{9}$$

$$y^t_{ijc} \in \mathbb{Z}^+ \quad \forall\, i\in\Phi,\, j\in\Phi, c\in\Omega,\, t\in\Psi \tag{10}$$

The objective function (3) has three terms: the cost of transporting products to the disaster areas; the cost of moving vehicles between depots, shipping origins and shipping destinations; the cost of carrying prepositioned inventory. The set of constraints (4) ensures the demand satisfaction. The constraint set (5) restricts the available transportation capacity for each period and site. The constraint set (6) controls that the number of transferred vehicles does not exceed its availability for each period. The constraint set (7) ensures that the flows of products do not exceed the initial inventory available to the suppliers. Constraint set (8) controls that the initial stock level does not exceed the inventory capacity. Finally, the constraint sets (9) and (10) impose integrality and non negativity. If it was not for the cumbersome constraint (4), this model could be solved through standard mixed integer programming methods. Unfortunately, the shape of (4) precludes that approach. Consequently, another technique must be used to incorporate the inherent stochasticity. One such approach is the Sample Average Approximation method described in the following section.

4 Sample Average Approximation

In this section, a Sample Average Approximation (SAA) scheme is implemented (as presented in [Pagnoncelli *et al.*, 2009]). The general idea is to replace the constraint (4) by a (larger) set of new deterministic constraints that approximate the stochastic model; it will be a larger model but simpler, because it does not consider stochasticity at all, nevertheless it must be solved repeatedly to simulate the probability of event occurrence considered in the original model, which increases the complexity of the problem. The SAA scheme implemented in this research solves a series of scenarios consisting in different instances of the modified optimization model (expressions 11 to 20), each instance corresponding to a realization of a Monte Carlo simulation of demands (expression 1). The solution of this series of instances of the modified optimization model provides a lower and upper bound to the solution of the original problem.

Let z^{tn}_{kp} be binary variables that measure the number of times that a demand constraint is not satisfied. Thus, the following modified optimization model is defined for the generated samples.

$$\min \sum_{n=1}^{N}\sum_{t\in\Psi}\sum_{p\in\Pi}\sum_{k\in\Phi}\sum_{j\in\Phi}\sum_{i\in\Phi} cv^t_{ijkp}x^{tn}_{ijkp} + \sum_{n=1}^{N}\sum_{c\in\Omega}\sum_{j\in\Phi}\sum_{i\in\Phi} cl^t_{ijc}y^{tn}_{ijc} + \sum_{t\in\Psi}\sum_{p\in\Pi}\sum_{j\in\Phi} ci^t_{jp}I^t_{jp} \tag{11}$$

s.a.

$$\sum_{j\in\Phi}\sum_{i\in\Phi} x^{tn}_{ijkp} + z^{tn}_{kp}D^{tn}_{kp} \geq D^{tn}_{kp} \quad \forall\, k\in\Phi,\, p\in\Pi,\, t\in\Psi,\, n=1,\ldots,N \tag{12}$$

$$\sum_{p\in\Pi}\sum_{k\in\Phi}\sum_{j\in\Phi} w_{pc}x^{tn}_{ijkp} \leq u_c \sum_{j\in\Phi} y^{tn}_{ijc} \quad \forall\, i\in\Phi,\, t\in\Psi,\, c\in\Omega,\, n=1,\ldots,N \tag{13}$$

$$\sum_{j\in\Phi} y^{tn}_{ijc} \leq V^t_{ic} \quad \forall\, i\in\Phi,\, t\in\Psi,\, c\in\Omega,\, n=1,\ldots,N \tag{14}$$

$$\sum_{k\in\Phi}\sum_{i\in\Phi} x^{tn}_{ijkp} \leq I^t_{jp} \quad \forall\, j\in\Phi,\, p\in\Pi,\, t\in\Psi,\, n=1,\ldots,N \tag{15}$$

$$I^t_{jp} \leq L_{jp} \quad \forall\, j \in \Phi,\, p \in \Pi,\, t \in \Psi \tag{16}$$

$$\sum_{n=1}^{N} \sum_{t \in \Psi} \sum_{p \in \Pi} \sum_{k \in \Phi} z^{tn}_{kp} \leq N(1-\gamma) \tag{17}$$

$$x^{tn}_{ijkp}, I^t_{jp} \in \mathbb{R}^+ \quad \forall\, i \in \Phi,\, j \in \Phi,\, k \in \Phi,\, p \in \Pi,\, t \in \Psi,\, n = 1, \ldots, N \tag{18}$$

$$y^{tn}_{ijc} \in \mathbb{Z}^+ \quad \forall\, i \in \Phi,\, j \in \Phi, c \in \Omega,\, t \in \Psi,\, n = 1, \ldots, N \tag{19}$$

$$z^{tn}_{kp} \in \{0,1\} \quad \forall\, k \in \Phi,\, p \in \Pi,\, t \in \Psi,\, n = 1, \ldots, N \tag{20}$$

where, the upper index índice n is the sample number, N is the sample size and γ is the desired level of accuracy to solve the approximated problem. this level of service is not necessarily identical to the original α level originally defined. Constraint (17), allows that the number of times that the demand satisfaction constraint (12) is violated , does not exceeds $1-\gamma$.

4.1 Lower Bound

To obtain a lower bound a sample average approximation scheme is applied (see [Pagnoncelli *et al.*, 2009]). The first step is to find two integer numbers M and N such that:

$$\theta_N := \sum_{i=0}^{\lfloor (1-\gamma)N \rfloor} \binom{N}{i} \alpha^i \left(1-\alpha\right)^{N-i} \tag{21}$$

and L being an integer number such that:

$$\sum_{i=0}^{L-1} \binom{M}{i} \theta_N^i \left(1-\theta_N\right)^{M-i} \leq 1-\beta \tag{22}$$

Then, a set of M independent samples must be generated: $D^{t1m}_{kp}, \ldots, D^{tNm}_{kp}, m = 1, \ldots, M$ each one of size N.

For each generated sample, the above modified optimization problem must be solved.

The optimal solution for each sample, called $\hat{\theta}^m_N$, $m = 1, \ldots M$, must be arranged in non decreasing order, $\hat{\theta}^{(1)}_N, \ldots, \hat{\theta}^{(M)}_N$, where $\hat{\theta}^{(i)}_N$ is the i-th smallest value.

Finally, the value $\hat{\theta}^{(L)}_N$ will be a lower bound for the optimal solutyion of the original problem, with a significance level of at least β.

4.2 Upper Bound

To obtain an upper bound, the method put forward by [Luedtke and Ahmed, 2008] will be applied. One of the findings of that article is the size N of a sample to guarantee that the solution of the modified optimization problem be in fact a feasible solution for the original problem, with a significance level of β. The latter is obtained as follows:

$$N \geq \frac{2}{(\alpha-\gamma)^2} \log\left(\frac{1}{1-\beta}\right) + \frac{2m}{(\alpha-\gamma)^2} \log\left\lceil \frac{2DL}{\alpha-\gamma} \right\rceil \tag{23}$$

This result gives a theoretical guide for the search of the sample size N. However, the problem size (given the obtained value of N) could be prohibitively large.

An alternative to this method is to solve the modified optimization problem with a smaller value of N and then checking (a porteriori) the fulfilment of the stochastic constraint.

This a porteriori checking can be done by using a sample of size N', and then for the samples $D^{t1}_{kp}, \ldots, D^{tN'}_{kp}$ counting the number of times that this expression holds: $\sum_{j \in \Phi} \sum_{i \in \Phi} x^{tn}_{ijkp} \geq D^{tn}_{kp}$.

The upper bound will be the objective's function value, corresponding to the solution with the highest value within all the feasibel solutions, once the a posteriori checking was performed (see [Luedtke and Ahmed, 2008]).

5 Numerical Example

In this section a numerical instance is presented, applying the above described methods to obtain both the lower and upper bounds.

The example consists of an instance with six locations, four periods, two products and two types of vehicle. Each location faces an iid lognormally distributed demand with mean 100 and a standard deviation of 10. For each period, the probability of a natural flood is assumed to be 0.2, with spatial and temporal correlations generated randomly.

5.1 Lower Bound

Using the methodology presented in section (4.1), the relevant parameters were found: $M = 172$, $N = 20$, $L = 12$, considering the following values: $\alpha = 0.9$, $\beta = 0.99$, $\gamma = 1$.

Using CPLEX 12, the modified optimization problem (presented in section 4) was solved $M = 172$ times. Each instance consisted of 960 binary variables, 5,760 integer variables and 34,608 continuous variables. Demands were generated with a Monte Carlo simulation routine using the parameters given above.
The 172 obtained solutions were then sorted in a non-decreasing order, obtaining a lower bound in $L = 12$, which value corresponds to $47,392$.

5.2 Upper Bound

To obtain the upper bound, we considered the values $N = 30$ and $\gamma = 0.95$, obtaining a feasible solution, which was tested a posteriori, considering 100 randomly generated demands. The instance consisted in 1,440 binary variables, 8,640 integer variables and 51,888 continuous variables.
The solution found was feasible in all of the 100 considered demands, which is a successful result given the high level of confidence set initially: $\alpha = 0.9$. The optimal value for each one of the 100 instances is the following. Minimum value = $52,989$; Maximum Value = $67,124$; Average = $55,991.6$; Standard deviation = $4,601.4$.

Thus, the upper bound to the original problem corresponds to $67,124$.

5.3 Results Analysis

The difference between both bounds is 41.6%. This gap is larger than that of other published results [Luedtke and Ahmed, 2008]. However, the type of problem addressed in this case is rather different from those previously tackled in the published literature. Indeed, natural floods are low probability-high consequence events, which means that the stochastic parameters exhibit a high variability when compared to other problems; in this case the coefficient of variation is 2, which is considerably higher than those in the reviewed literature (between 0.1 and 0.5 depending on the case).

Even though there is no formal indication about how conservative both bounds are, we can conclude, with high level of confidence, that the solution found for the upper bound will satisfy the emergency demands with the desired significance level. This is, of course, a highly valuable input to the decision makers, in spite of not knowing the exact value of the optimal objective function.

6 Conclusions

The occurrence of a flood is an event of low probability but often of high consequences in terms of material costs and sometimes human or animal losses. These characteristics (along with the size of realistic instances) make that phenomenon unsuitable to be modeled with standard mathematical programming techniques, which assume deterministic parameters.

In this article, the authors presented a stochastic programming model to represent tactical decisions in the logistics of emergencies after the occurrence of a natural flood. The model attempts to find the optimal levels of inventory in different locations as well as the flows of materiel from these locations to the affected areas. The model also gives the fleet size that must be available at each location to distribute an array of different types of products (which may need different types of vehicle to be delivered). In addition, the authors proposed a methodology to find an approximation to the optimal solution of that model. The approximation is found through the application of a recently published scheme of sample average approximation.

The proposed model and solution approach are general in the sense that no specific probability distribution function is assumed (or necessary) and the pattern of spatial and/or temporal correlations can be totally general. The applicability of this modeling structure is conditioned on the capacity of the modeler to generate spatiotemporal patterns similar to those observed in the real scenario as well as the capacity to handle the large size of the resulting instances to be solved.

The modeling structure was applied to a test instance with six locations, two prodcuts and four periods. The spatial and temporal correlations between flood-occurrence probabilities were generated randomly. Through the described methodologies upper and lower bounds were found for the original optimization problem. The gap between both bounds was relatively large (41.6%). However, the solution for the upper bound satisfies the emergency demands with the desired level of confidence, which is a very useful result for the decision makers under the preassure of delivering emergency aid in extreme events.

Acknowledgments

The authors would like to express their gratitude to FONDECYT through Grant 1080189, as well as the partial funding by NSF project DRU: contending with materiel convergence: optimal control, coordination, and delivery of critical supplies to the site of extreme events, with Principal Investigator José Holguin-Veras.

References

M.S. Chang, Y.L. Tseng, and J.W. Chen. A scenario planning approach for the flood emergency logistics preparation problem under uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):737–754, 2007.

F. Fiedrich, F. Gehbauer, and U. Rickers. Optimized resource allocation for emergency response after earthquake disasters. *Safety Science*, 35(1-3):41–57, 2000.

K.S. Kelly and R. Krzysztofowicz. Probability distributions for flood warning systems. *Water Resources Research*, 30(4):1145–1152, 1994.

J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

L. Ozdamar, E. Ekinci, and B. Kucukyazici. Emergency logistics planning in natural disasters. *Annals of Operations Research*, 129(1):217–245, 2004.

BK Pagnoncelli, S. Ahmed, and A. Shapiro. Sample Average Approximation Method for Chance Constrained Programming: Theory and Applications. *Journal of optimization theory and applications*, 142(2):399–416, 2009.

Re-organization in Warehouse Management Systems

Huib Aldewereld, Frank Dignum, and Marcel Hiel
Utrecht University - Institute of Information and Computing Sciences
Utrecht, The Netherlands, {huib,dignum,hiel}@cs.uu.nl

Abstract

Warehouse Management Systems (WMS) are traditionally highly optimized to a specific situation and do not provide the flexibility required in contemporary business environments. Agents have been advocated for their flexible and adaptive nature, but require organizational structure to ensure that the system performs as required. Due to changes in the environment a different organization may be more productive making re-organization essential. In this paper, we present an architecture and methodology for easing the redesign of a WMS. The method applied is based on heuristics for re-organization given the environment, the main objectives of the organization and the current situation.

1 Introduction

Warehouse Management Systems (WMS) are traditionally centralized, monolithic software systems that are highly optimized for a specific situation. These systems thus guarantee very efficient operation given some fixed constraints. However, these systems usually have trouble to achieve flexibility (such as handling priority orders) and robustness (such as machine failures). In modern WMS a good balance between efficiency, flexibility and robustness is of utmost importance.

Agents were introduced to tackle the problem of flexibility (e.g., see [2; 7]). However, only introducing agents is not sufficient. An argument against the usage of agents is that the control is hard to guarantee and that therefore the requirements of robustness and efficiency cannot be guaranteed. The main problem is that efficiency, flexibility and robustness are aspects that pertain to the system as a whole. Moreover, we cannot optimize all three at the same time, but have to balance the three aspects (e.g., more robustness usually means less efficiency). Thus, when an agent approach is used it does not mean that each agent has to manage the balance between the three aspects. However, the distributed nature of this type of solution makes it very hard to guarantee an overall balance between the three aspects. In order to prevent anarchy and regulate the agents within such a system, we propose to use agent organizations [6]. An agent organization is used to specify exactly those aspects of the system that need to be guaranteed by the agents together. Individual agents can have their own goals and ways of interacting with other agents. However, because they are designed to fit the agent organization their autonomy is limited by the overall objectives of the organization. E.g., suppose that we have 4 types of tasks occurring equally frequent and four machines that can perform all tasks. For efficiency sake we probably want each machine to specialize in one type of task (which might avoid reset time etc.). However, for robustness sake we want all machines to perform all tasks. We could choose a balance of having two machines perform two tasks such that if one fails the other takes over and work on that task does not completely halt. How the machines subsequently divide their tasks they might decide themselves (using agent based solutions for this situation).

Unfortunately, although organizations provide structure and stability for regulating a multi-agent system (MAS), the environment might change in ways such that the organization no longer guarantees the right performance. For example, after the introduction of new hardware or product changes, the WMS may provide a suboptimal solution. In order to fully use the flexibility provided by agents and maintain the robustness of an organization, being able to re-organize is essential.

However, in order to perform a successful re-organization, it should be done at the right moment and changing the organization in a way to perform better. In this paper we will show how agent organizations can be used to implement a WMS and how successful re-organizations can be carried out in this environment.

The alternative to re-organization is to make the agents themselves adaptive. The distinctive difference between adaptive agents and re-organization is that in the organization the knowledge concerning the global (organization-wide) objectives is explicit. The advantage of having this knowledge explicit is that it is easier to adjust when modification is necessary.

This paper is structured as follows: In Section 2 we first introduce a motivating example. Subsequently we show how agent organizations are used to control the warehouse in Section 3 and which criteria are used to measure its success. In Section 4 we show how re-organizations can be defined using change patterns and illustrate how they are used to keep the organization successful under changing circumstances. We conclude in Section 5 with some observations on our use case and future work.

2 Motivating Example Scenario

A warehouse stores and collects products for customer orders. These products are typically packed and/or placed in boxes or containers, generally referred to as Transport and Storage Units (TSU). Figure 1 illustrates our example hardware configuration for a WMS. This figure contains three types of components, namely miniloads, conveyorbelts and workstations. The miniloads are storage units where TSUs are kept, the conveyorbelts are responsible for transporting TSUs between miniloads and workstations, and the workstations represent places where operators pick products from TSUs for fulfilling the orders. TSUs are not kept at the workstations, but requested from the miniloads when an order arrives and returned immediately after picking the required amount of products to fulfill the order. In our configuration, thus, we have three miniloads, one conveyorbelt and two workstations. The boxes on the conveyorbelt represent moving TSUs.[1] The squares with arrows represent buffers where the direction of the arrow indicates the direction of movement.

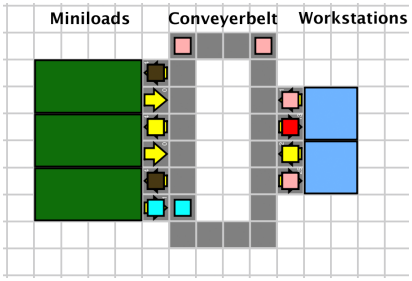


Figure 1: Example warehouse configuration

Assume that this WMS performs adequately. However, due to a successful marketing campaign one particular product becomes very popular. At a certain point, all components work at optimal efficiency, however, because of that popular product the number of orders becomes larger than can be handled per day and therefore the number of outstanding orders becomes larger and larger. Moreover, because of this delay customers become unsatisfied due to the long waiting time.

In order to deal with this situation management decides that the warehouse should be extended, however due to limited space only two workstations can be added but no miniloads. As the average miniload is calculated to be able to support one or two workstations (in processing time), expectations are that number of orders handled will improve. After placing the workstations the performance (throughput) increases but not as much as expected and the number of outstanding orders still increases.

In the next section we will first describe the basic set-up of the agent organization that controls the warehouse logistics and how this supports the balance between efficiency, robustness and flexibility of the WMS. In the rest of the paper we will show how the changing environment renders the organization inefficient. In order to determine whether the performance can be increased, re-organization is to be considered, which is explained in the sections after that.

[1] The color of the TSUs indicates the product family, which is unimportant in the discussion in this paper.

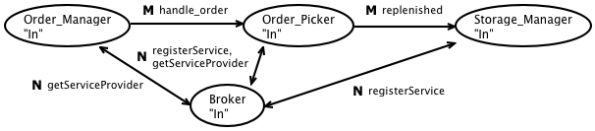


Figure 2: Organization for Planning

3 Agent-organizations in Warehouse Management Systems

In [6] we showed how agent organizations are used to balance flexibility, robustness and scalability. Here we only highlight the most important aspects of this agent organization model. In our architecture, we use agents to control every component (thus miniload, conveyorbelt and workstation). Every agent is responsible for, and optimizes the efficiency of, its component. The agent organization structures the interactions between agents and provides answers to design questions, such as: who talks to who?, what is the role of the agent within the organization?, and what are the objectives that this agent seeks to achieve?

In the remainder of this section, we describe our layered approach for modeling agents after which we present performance indicators that can be used to evaluate a warehouse management system.

3.1 Layers

Warehouse management systems are typically thought of in three layers of operation, namely the plant (execution), scheduling and planning. This distinction in layers is made to create a separation of concerns which makes it easier to create a modular design and thereby support decoupling of the different aspects of the WMS.

This separation is reflected in our model (for details see [6]) based on the MASQ meta-model [8]. Each of the layers of the warehouse management system, that is, the plant, scheduling and planning layers, correspond to a separate interaction space which defines the particular protocols that the agents use on that layer. Because each space incorporates its own implemented business rules and interaction protocols this improves modularity and maintainability.

Planning Space: "Planning is the process of generating (possibly partial) representations of future behavior prior to the use of such plans to constrain or control that behavior" [1]. In our domain this means that orders are assigned to be handled by certain components (without an explicit timing). The interaction necessary for the assignment of orders is modeled as an agent organization using the OperA framework [3; 4].

The social structure of the agent-organization, which specifies the roles and the relations between these roles, is shown in Figure 2. The arrows between the roles indicate dependency relations. From the figure it is clear that there is a role taking care of incoming orders and that the agents fulfilling the "Order_Picker" role (in our case the workstations) will

Presence	Stock Management	Broker Communication
Order_Picker	createReplenishCFP	registerDeliveryService
	createDeliveryProposal	requestReplenishProviders
Storage_Manager	createReplenishProposal	registerReplenishService
Order_Manager	createDeliveryCFP	requestDeliveryProviders

Table 1: Planning Capabilities per Goal and Presence

Keeping plant active	Scheduling Transfer of TSUs	For Planning
getNextAction	handleIncomingMessage	getLeadTime
processAction	sendTSUPlacementRequest	scheduleTSU
isOutgoingTSUScheduled	sendTSUPlacementReply	
handleIncomingTSU		

Table 2: Scheduling Capabilities per Goal

provide the order. In order for the Order_Picker to get the necessary products for the order they ask the agents fulfilling the "Storage_Manager" role (in our case the miniloads) to provide the products from storage. Finally the Broker maintains knowledge about which roles exist in the system and which service(s) they can provide.
Through this organization structure we already convey that the incoming orders determine the logistics (we did not incorporate a product reception role). We also do not connect the customers directly with the storage manager. This means that the order picker decides whether an order should be processed and never the storage manager. The underlying reason to model it this way is that the workstations are known to form a bottleneck. We therefore want those to be in charge of the workload. If they are optimally used the whole system performs optimally.
Besides these basic points we can also get information from the types of dependencies between the roles. There are different types of dependencies possible, each resulting in a different type of interaction; bidding [**M**arket], delegation [**H**ierarchy], and request [**N**etwork]. The protocol that is used between the agents depends on the kind of interaction type specified in the organization model. In our case, market relations are implemented using the Contract Net Protocol, and network relations are implemented as request/inform messaging. The use of market relations and the Contract Net Protocol as implementation for service requests means that, given an appropriate bidding mechanism, balance of the workload of the components is achieved automatically. Thus we provide for robustness and flexibility within the organization structure. Note that the efficiency of the resulting logistic process depends on the decision mechanisms of the agents.
Another aspect that can not directly be seen from Figure 2 is that all miniloads can communicate with all workstations, but they do not communicate amongst each other. This makes the configuration very robust and flexible, but potentially less efficient. It also contributes to an quadratic growing amount of communication.

The required capabilities of the agents on planning are summarized in table 1.

Scheduling Space: In our domain scheduling encompasses three goals, namely (1) supplying the hardware with actions to perform, (2) transferring TSUs from one component to another, and (3) provide information and the means for planning such that plans can be created and executed. For each of these objectives a number of capabilities are required. Table 2 lists the capabilities that we distinguish.

As can be seen from the above, in our (simple) scenario scheduling has little independence from planning and thus we do not provide the social structure for this part as it is completely in line and enforcing the objectives of the planning. Mainly, scheduling ensures that the hardware, at all times, knows what to do next. After the plant gives a notification that the current action is finished, the scheduling component of the appropriate agent supplies it with the next action (getNextAction). Furthermore, scheduling maintains a list of TSUs in the component, as well as those that are planned for movement (handleIncomingTSU and isOutgoingTSUScheduled). Maintaining this list after executing an action is done by the processAction capability.

As planning is dependent on information from scheduling, for example in calculating the time it would cost to process a TSU (leadTime), scheduling components provide capabilities to get this information, such as the getLeadTime capability. Furthermore, the scheduling components present the capabilities for scheduling a TSU to be processed by the plant.

3.2 Performance

Organizations have to try to achieve three global objectives, namely efficiency, robustness and flexibility. An organizational structure maintains a certain balance of these objectives with respect to its environment. However, if the environment changes the balance might shift into an unfavorable direction. The questions are then: what criteria should be measured in an organization in order to decide to re-organize and when to change? In general, efficiency can be linked to throughput (i.e., the amount of orders processed per day). Robustness is more difficult to measure. It should be measured by resilience to failure of machines. One can compare the throughput of the warehouse when one machine fails with the throughput of the warehouse where it is configured optimally without that machine. Finally flexibility depends on environment parameters. In this case one compares the performance of the warehouse in case all events were known on forehand with the case where some events happen unexpectedly.

It is clear that the measurements get more and more difficult when going from efficiency to robustness to flexibility. E.g., a warehouse might be very robust with respect to failing miniloads but very sensitive to failing workstations. In practice one looks at one or more bottlenecks in the warehouse and checks the robustness with respect to their failure. The same holds for flexibility. A warehouse might be very good at handling priority orders as long as the orders required 'standard' products. However, it may perform poorly when the priority orders require combinations of products that are rare. Again, in practice one only measures those cases that are expected.

Because in our scenario overall efficiency of the warehouse is the most important objective we use simple throughput as the performance criterium. Figure 3 illustrates a graph of the throughput for our running example. The figure contains four

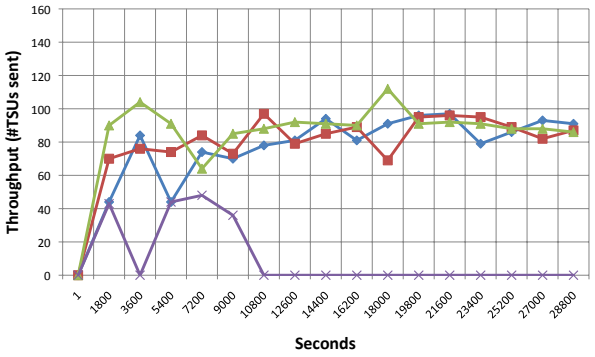


Figure 3: TSU Throughput for four workstations

lines, one for each workstation. One of these four lines climbs in the beginning, but then quickly drops to zero. After placing the two additional workstation the miniloads are not quick enough to handle all the requests on time, therefore the fourth workstation is without work most of the time. It is clear that the organization is not performing optimally in the new configuration. In the next section we will investigate what type of changes we can make to the organization in order to improve the performance.

4 Developing Change Patterns for Re-organization

It is impossible to know all the changes that may affect a warehouse in advance. Therefore an exhaustive overview of changes and corresponding action for adaptation is hard to provide. However, we can provide some general heuristics for each of the main warehouse objectives: efficiency, robustness and flexibility. We will first give these general heuristics and subsequently describe in more detail how the re-organization of our scenario is realized.

Efficiency When throughput should be increased first one finds the bottleneck components. Basically the agents indicate their performance as a percentage of their maximal performance. The agents with the highest percentage will be the bottleneck. The next step is to add more components of that type (e.g., in our scenario we will add more workstations). Now two steps should be taken with respect to communication. First it should be checked whether the components fulfilling the same role were communicating amongst each other already. If not, it should be checked whether they should start such communication now. This communication is added when the products leaving one component might be interesting for another component and thus a kind of ”sideway” logistic step is added.
Finally, it should be checked whether a new role should be added in between other roles for communication efficiency. In our case all miniloads communicate with all workstations. If there are too many of each type each component is all the time communicating and processing information about possible work, most of which would be useless. In that case a new intermediary can be created that makes the decision on division of work based on communication with both sides (we reduce the amount of communication channels from $n * m$ to $n + m$).

Robustness When robustness has to be increased we have to create alternative potential workflows. There are two ways to create alternative workflows. The first is by adding more components of a certain type. This is handled in the same way as indicated above for increasing efficiency. The second way to create alternative workflows is to connect more agents fulfilling different roles. In the extreme case every agent and associated component is connected to every other agent/component. Of course many connections are useless, because the associated components cannot exchange their products (in a meaningful way), but it guarantees that every possible workflow is indeed covered by the organization. Our example warehouse looks very robust, because all miniloads are connected with all workstations. However, they all use the same conveyorbelt. To make things more robust separate connections could be used for every pair of miniload and workstation. The decision to create such alternative paths is made on the basis of the expected rate of failure of each component and the costs to create and maintain an alternative path.

Flexibility To increase flexibility the general approach is to create components with more capabilities. In this way each component is better capable to handle more situations. In the warehouse domain a relatively cheap way to increase the capability of a component is to create an input and output buffer. This creates the possibility for the component to have more flexibility on deciding what task to perform next (of course this is a limited form of flexibility, but an often occurring need). Once components get more capabilities it is also possible to create more alternative workflows, thus increasing the robustness of the system.
When components have more capabilities, of course, they also need to have the decision mechanism on how and when to use the capabilities. Besides that they need the right information to make the right decisions. In general this means that new communication channels have to be created to get the information to all the components that need it. In the most extreme case all components can perform all tasks and exchange all possible information with all other components. This is clearly not very cost efficient, but extremely flexible and robust also.

In general the organization should balance efficiency, robustness and flexibility and all the required levels should be attained at a minimum cost (in terms of resources, communication, and complexity). A re-organization makes sense if the added expected gain in performance of the system is higher than the additional costs. Here “performance” is meant in a wider sense than just throughput, but rather behaviour of the system with respect to all criteria over a period of time. Crudely stated: if the gain in profit by the re-organization is higher than the costs (calculated over a certain period of time), re-organization makes sense.

Given these general heuristics for re-organization, we now turn to the concrete example. We describe how to capture the experiences of redesign such that they can assist developers in future projects. In particular, we focus on change patterns. Change patterns, and design patterns in general, provide for a structured documentation thereby making the transference

	Concept	Operator	Description
Organization	Role	addRole(r_i)	add role r_i
		removeRole($name$)	remove role r_i
	Dependency	addDependency(d_i, r_{from}, r_{to})	add dependency d_i between role r_{from} and role r_{to}
		removeDependency(d_i)	remove dependency d_i
	Objective	addObjective(o_i, d_i)	add objective o_i to dependency d_i
		removeObjective(o_i, d_i)	remove objective o_i from dependency d_i
	Player	addPlayer(p_j, r_i)	add player p_j to perform role r_i
		removePlayer(p_j)	remove player p_j
Agent	Agent	addAgent(a_i)	add agent a_i
		removeAgent(a_i	remove agent a_i
	Presence	addPresence(p_i, a_i, l_j)	add presence p_i to agent a_i for layer l_j
		removePresence(p_i, a_i)	remove presence p_i from agent a_i
	Capability	addCapability(c_k, p_i, a_i)	add capability c_k to presence p_i of agent a_i
		removeCapability(c_k)	remove capability c_k from presence p_i

Table 3: Change Operators

Name	Communication Between Pickers
Change Type	Re-organization of planning layer
Trigger	A popular product & nr orders received > nr orders handled (per day)

Change Template	//Organizational model
	addDependency(d6,order_picker,order_picker);
	addDependencyObjective(replenish,d6);
	
	//Planning Implementation (for all players p)
	//to PickerPlannerPresence
	addCapability(registerReplenishService,getPlanBody(p),p);
	addCapability(createReplenishProposal,getPlanBody(p),p);
	
	//Scheduling Implementation
	addCapability(scheduleTSU,getScheduleBody(s),s);

Table 4: Change Pattern Communication Between Pickers

of knowledge between developers easier. Making this knowledge explicit thereby reduces the time required for (re-)design and (re-)implementation.

4.1 How to re-organize?

Following a model-based approach, we use models to get an overview of what can be done. More specific, we use a model-management approach that was used to describe the evolution of services [5]. In this model-management approach, with a *model*, a complex structure is meant that represents a design artifact. The usage of models implies manipulation and transformation of one model to another model. The key idea behind model-management is to develop a set of algebraic *operators* that generalizes the transformation operations. In our framework, these operators consist of adding and removing concepts (and relations) in a model. Operators are commonly stored in a script. A script is a sequence of operations that (automatically) transforms one model into another.

In our approach, we have two types of models, namely the organizational model, and a model which represents the implementation of the agents. We list the operators for each of these models in Table 3. These change operators represent all the possibilities for changing the organization and the agents. For example, if the organization grows substantially by incorporating many more agents, a manager role can be introduced through the addRole operator. As the organizational model is a specification, the operators used for this model can be automated. In other words, a new organizational model can be automatically derived by applying these operators to the old model.

Next to the organization model, the agent(s) implementation should also be changed to reflect the new organization. However, this might imply giving agents new goals, new protocols, etc. Because we do not desire to restrict the agent implementation we only require the agents to incorporate the concepts in Table 3. Here we provide only guidelines and requirements for how to structure the implementation such that these operators can be used to precisely determine where to update the implementation.

Following our model of an agent, we can either add or remove presences for agents in the different spaces. Furthermore, capabilities can be added to or removed from the different presences at both planning and scheduling. Note that these capabilities provide only the skeleton functionalities and that they still have to be programmed.

In order to enhance reusability of existing code and thereby reduce the time to implement the new WMS, the addition of new capabilities should be done mainly based on the existing capabilities. That is, the operators that are specified should preferably exist in other presences, such that the developer already has an idea of what the capability should do and what decisions are important for this capability.

4.2 Example Scenario: Communication Between Pickers

We return to our running example scenario. With the additional workstations the number of orders handled becomes larger but the new configuration did not solve the complete problem. The solution found by the warehouse manufacturer is that TSUs containing the popular product need not be returned to the miniload but can be send directly between workstations if needed. This cuts the traveling time back to the miniload as well the processing time of the miniload for getting the TSU out of the rack.

This change requires a re-organization where workstations do not only communicate with the miniloads, but also communicate with other workstations in order to request and propose replenishments. The pattern for this situation is shown in Table 4. The change template shows the operators needed for creating the new WMS. The first two operators apply to the organizational model and can be automatically applied. The other operators affect the implementation and cannot be automatically applied. These operators add capabilities to presences of agents on both the scheduling and planning space. On the planning space the capability should be added that pickers register themselves as replenish service providers at the broker. This causes them to be contacted if components (pickers) require replenishment. Furthermore, the createReplenishProposal should be added such that pickers can create proposals to answer to the call-for-proposal (cfp) of the Contract Net Protocol. In the scheduling space the TSU should be scheduled to go to a picker instead of back to the miniload. Therefore the capability should be added to schedule the TSU.

In our change pattern, the capabilities can be reused from other components. For example, the createReplenishProposal is already a capability of the miniloads, however, the reason-

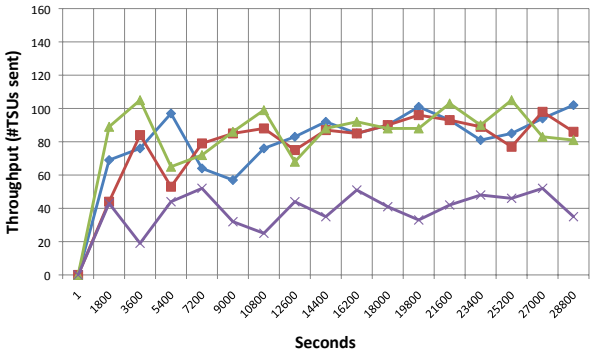


Figure 4: TSU Throughput with picker communication

ing to decide whether a TSU is available differs whether it is in the miniload or in a picker.

The result of applying this change pattern is shown in Figure 4. As can be seen when comparing Figures 3 and 4, the fourth workstation increased its throughput by roughly 40 percent, due to the communication between pickers. This shows that the problem has been solved using the heuristic mentioned in the beginning of this section. First the management located the bottleneck of the old warehouse (the workstations) and added more components of that type. This increased performance slightly. Second, the flow of product TSUs was improved by enabling horizontal communication between the pickers. Note also that the robustness of the warehouse went up as well (but only with respect to situations where workstations fail).

5 Conclusion

Although in Warehouse Management Systems efficiency is of prime concern it is clear that this has to be balanced with robustness and flexibility of the system. We have shown how agent organizations can be used to balance efficiency, robustness and flexibility of a system. However, due to changing circumstances in the environment an organization might not keep the right balance over time. Thus re-organization of agent-organizations is essential in an evolving environment. In order for the re-organization to have a positive effect we first have to check which criteria could be used to measure the performance of the organization with respect to each of the aspects. For efficiency this is reasonable unique and can be captured by throughput given a set of resources. Robustness and flexibility have to do with responses to resource and communication failures and "unexpected" events. So, the criteria should actually be detailed with respect to the kind of failures and "unexpected" events. We have given some general criteria and subsequently described some general heuristics for re-organization scripts based on these criteria. In the example scenario it is clear that when the throughput of some of the newly added workstations is almost zero the organization is performing badly.

The heuristics that were used to change the Warehouse organization had an effect on organization efficiency; if a role forms a bottleneck, additional players of that role can be added to distribute the load. This change, however, shifted the bottleneck to the miniloads, which did not perform optimally due to lack of coordination between them. Several things could have been done to remedy the situation. The first would be to add additional miniloads, but this is costly (and not possible given the space limitations in our scenario). Secondly, a manager role could have been added to the pickers to streamline the distribution of work, and allow for the reuse of replenishment TSUs among the pickers. This would, however, add another layer in the organization (complicating communication by adding an extra step in the chain) while the decisions to be taken by that manager are relatively simple (and can be taken by the miniloads themselves). Therefore the option we chose (which is less drastic and more flexible) was to allow for horizontal communication between the picker agents. We saw that this change could be captured by change patters that guide the re-organization process. Of course, the next steps will be to further investigate heuristics that guarantee at least a certain amount of improvement in efficiency, robustness, and/or flexibility while keeping a right balance given the changing environment. And we will test our heuristics systematically in many different types of scenarios.

Acknowledgment

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

References

[1] Austin Tate in the MIT Encyclopedia of Cognitive Science. Planning. `http://cognet.mit.edu/library/erefs/mitecs/tate.html`.

[2] R.S Chen, K.Y. Lu, and C. C. Chang. Intelligent warehousing management systems using multi-agent. *Int. J. Comput. Appl. Technol.*, 16(4):194–201, 2003.

[3] Virginia Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Universiteit Utrecht, 2004.

[4] Virginia Dignum and Huib Aldewereld. OperettA: Organization-oriented development environment. In *LADS2010@MALLOW*, 2010.

[5] Marcel Hiel. *An Adaptive Service-Oriented Architecture - Automatically solving Interoperability Problems*. PhD thesis, Tilburg University, September 2010.

[6] Marcel Hiel, Huib Aldewereld, and Frank Dignum. Modeling warehouse logistics using agent organizations. In *CARE' 10*, LNCS. Springer-Verlag, to appear.

[7] Teruaki Ito and S. M. Mousavi Jahan Abadi. Agent-based Material Handling and Inventory Planning in Warehouse. *Journal of Intelligent Manufacturing*, 13:201–210, 2002.

[8] Tiberiu Stratulat, Jacques Ferber, and John Tranier. MASQ: Towards an Integral Approach to Interaction. In *AAMAS '09*, pages 813–820, 2009.