PROCEEDINGS

3RD WORKSHOP ON

ARTIFICIAL INTELLIGENCE AND LOGISTICS

(AILOG-2012)

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

August 28, 2012

20th European Conference on Artificial Intelligence (ECAI)

Montpellier, France

3RD WORKSHOP ON ARTIFICIAL INTELLIGENCE AND LOGISTICS (AILOG-2012)

ORGANIZERS

Lutz Frommberger
Cognitive Systems Research Group, University of Bremen, Germany
lutz@informatik.uni-bremen.de

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

Bernd Scholz-Reiter
BIBA - Bremer Institut für Produktion und Logistik GmbH, Bremen, Germany
bsr@biba.uni-bremen.de

PROGRAM COMMITTEE

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)
Stefan Kirn (University of Hohenheim, Germany)
Alexander Kleiner (Linköping University, Sweden)
Herbert Kopfer (University of Bremen, Germany)
Andreas D. Lattner (University of Frankfurt, Germany)
Martin Lauer (Karlsruhe Institute of Technology, Germany)
Jacek Malec (Lund University, Sweden)
Hedda Schmidtke (Carnegie Mellon University, Rwanda and USA)
Jaime Sichman (Universidade de São 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

BIBA - Bremer Institut für Produktion und Logistik GmbH
Bremen, Germany
http://www.biba.uni-bremen.de

TABLE OF CONTENTS

PREFACE

AUTONOMOUS AND DECENTRALIZED APPROACHES IN LOGISTICS

With AILog-2012 the workshop "Artificial Intelligence and Logistics" will be held for the third time. From the submitted manuscripts we selected 8 papers for presentation at the workshop after a thorough peer-review process. As in the previous years we could attract authors covering a wide range of problems and solution methods. This again shows the difficulty of modern Logistics problems. There certainly is a need for powerful solution methods, such as AI methods, in order to successfully cope with the complexity and requirements of current and future logistic systems and processes. In our opinion, especially decentralized and autonomous approaches seem to be very promising. In the presented papers this theme is taken up by many of the papers concerned with supply chain scenarios. An inherent geographical as well as organizational distribution of such processes seems to naturally match the use of decentralized methods such as multi-agent systems.

We want to thank all the authors for their contributions and the members of the program committee and the external reviewers (Paul Karaenke, Thomas Makuschewitz, Fernando J. M. Marcellino, Michael Schuele, Steffen Sowade and Rinde van Lon) for the substantial and valuable feedback on the submitted manuscripts. Torsten Hildebrandt again provided significant help in the workshop organization. We thank collaborative research centers "SFB/TR 8 Spatial Cognition" and "SFB 637 Autonomous Cooperating Logistic Processes", funded by the German Research Foundation (DFG), for their support.

The AILog workshops aim at aggregating a 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 previous AILog workshops, we hope to achieve it with this year's workshop as well. We are looking forward to an inspiring exchange of ideas and fruitful discussions in Montpellier.

Lutz Frommberger
Kerstin Schill
Bernd Scholz-Reiter

(AILog-2012 organizers)

INVITED TALK

SMART FACTORIES AND THEIR IMPACT ON SMART LOGISTIC SYSTEMS

DETLEF ZÜHLKE, DFKI KAISERSLAUTERN, GERMANY

Factories will face major changes over the next decade. This change is characterized by the keyword ”smart factories”, i.e., the broad use of smart technologies which we face in our daily life already in future factories. More in detail this means that factories will benefit from the advances in computer sciences and electronics like cyber physical systems, wired and wireless networking and various AI techniques. The planning and control systems will change from today’s monolithic and hierarchical structures to more or less open networks with a much higher degree of autonomy and self-organization. Because of these fundamental changes this situation was described in Germany by a new paradigm ”Industry 4.0” characterizing the changes as the 4th industrial revolution. It is obvious that smart factories will also have a substantial impact on logistics which must fit into this new world.

In Kaiserslautern a large demo factory called ”SmartfactoryKL” was installed years ago in close cooperation with many industrial partners. This factory serves as a realistic testbed for developing and demonstrating new factory technologies. Close links to the German Research Center for Artificial Intelligence (DFKI) and also the local university allow for the necessary research actions and offer a unique environment for a beneficial transfer of the research results to industrial application.

This presentation will describe the experiences gathered by the Smartfactory consortium over the last years and identify the impact and challenges for future production systems.

Prof. Dr.-Ing. Detlef Zühlke received the MS in electrical engineering and computer sciences and his PhD in robotics both from RWTH Aachen/Germany. Currently he is a Professor for Production Automation at the University of Kaiserslautern and scientific director of the research department Innovative Factory Systems (IFS) at the German Research Center for Artificial Intelligence (DFKI). He is chairman of the executive board of the SmartFactoryKL, chairman of the IFAC CC 4 on Mechatronics, Robotics and Components and member of the IFAC Technical Board, advisory board member of the VDI/VDE-Society for Measurement and Automatic Control and member of the advisory panel of Springer publishing.

His research interest is in industrial control architectures, factory planning and operation and human- machine-systems for industrial applications.

TECHNICAL PAPERS

Finding Optimal Paths in Multi-modal Public Transportation Networks using Hub Nodes and TRANSIT algorithm.

Leonid Antsfeld[1] and **Toby Walsh**[2]

Abstract. We present a new algorithm to find optimal routes in a multi-modal public transportation network. This work is an extension of recent work on finding multi-criteria optimal paths in multi-modal public transportation networks [8] using the ideas of TRANSIT algorithm [4]. As a preprocessing step, we identify *hub stations*, a relatively small subset of all stations and then precompute optimal paths only between those *hubs*. Given a query between any two locations we show how to extract the optimal path in an efficient way using those hubs. In addition we present an improvement of our algorithm using service patterns. This allows us to significantly reduce both memory requirements and preprocessing time of previously reported algorithm by order of magnitude. Finally, we present results of our experiments on the Sydney metropolitan and the New South Wales state public transport networks.

1 Introduction

An Intermodal Journey Planner (IJP), is a computer system which can provide a traveler with an itinerary as a sequence of several modes of transport (bus, train, ferry, tram, metro, etc.). There exist many different systems that provide users with this kind of transit information, e.g. NSW TranportInfo [10], Google Transit [7]. Such systems are often available as a Web or smart phone application. The application asks a user to input an origin, destination and expected departure or arrival time and provides the user with recommended travel routes. Such systems are useful in encouraging people to switch from their private cars to use public transport services, thus reducing congestion, CO2 emission and providing the travelers a better experience.

We present here an algorithm to find an optimal route from a location A to a location B leaving at time t. This work is an extension of the idea described in [8]. First of all we observe that when one needs to change a service during his journey, there are only a relatively small number of stations where it is worth to do so. We call those stations *hub nodes*. Next we notice, that knowing how to get from one hub to another will allow us almost immediately to produce an optimal trip between any two (non hub) stations. In addition we describe another, new speed up technique, which brings substantial improvement in both memory requirements and preprocessing time, by precomputing optimal routes for a limited number of service patterns rather than for every particular service. We present comprehensive experiment results with real world public transportation network of scale of metropolitan city of Sydney (containing 9.3K multi-modal stations and 2.6M events) and the whole state of New South Wales (containing 46.5K multi-modal stations and 6.7M events).

[1] NICTA/UNSW, Australia, email: leonid.antsfeld@nicta.com.au
[2] NICTA/UNSW, Australia, email: toby.walsh@nicta.com.au

2 Related work

In recent years several algorithms have been developed that use precomputed information to obtain a shortest path in a road network in a few microseconds, [12]. However there has been less progress for public transport networks. In [2] H. Bast discusses why finding shortest paths in public transport networks is not as straight forward as in road networks. There are several issues that arise in public networks, which are not encountered in road networks. First of all public transport network are inherently multi-modal. i.e. the are at least two different means of transport. Other issues we need to consider are *time dimension*, *transfer time safety buffers*, *tickets cost*, *operating days*, etc. The recent and the most prominent result in this area is by H. Bast et al. [3]. They also use a notion of *hub stations*, but in completely different way. H. Bast et al perform Dijkstra searches from a random sample of source stations and choose as *hubs* stations those that are on the the largest number of shortest paths. They report a time of 10ms for station-to-station query for a North America public transportation network consisting of 338K stations and more than 110M events. Besides being relatively complicated, the main drawback of their algorithm is the large computational resources required for precomputation. The authors report requirements of 20-40 (CPU core) hours per 1 million of nodes.

The original TRANSIT algorithm [4, 5] is one of the best methods for finding shortest paths in very large road networks, [6], but was previously limited to a single mode of transport and static and undirected graphs. One of the advantages of the TRANSIT is that the final query time is completely indifferent to the path length, but only depends on the number of transit nodes of the origin and destination. Abraham et al. [1] formalized this property as a *highway dimension*. Intuitively a graph has small highway dimension if for every $r > 0$, there is a small set of vertices S_r such that every shortest path of length greater than r includes a vertex from S_r. The very impressive performance of TRANSIT on road networks suggests that road networks may have low highway dimension. On the other hand, the public transportation network, somewhat resembles the structure of road network, therefore it looked natural to extend and adjust the TRANSIT to public transportation network. In [8] we reported a promising direction extending the TRANSIT algorithm [4, 5] and applying it to an improved time expanded graph of the multi-modal public network. This approach is more intuitive, has much less hardware requirements and precomputation time. In addition we show how to provide multiple results in the real world incorporating user preferences.

3 Contribution

In this section we describe our new model of the public transport network and give a detailed description of our idea.

3.1 Modeling the Network

There are two main approaches to model a public transport networks, known as *Time-Dependent* and *Time-Expanded* models. For an exhaustive description of the models and existing techniques we address the readers to [9, 11]. We suggest a new model which is an enhanced combination of the two. Our model consist of two layers: *station graph* and *events graph*. The *station graph* nodes are the stations and it has two types of links *station links* and *walking links*. In our experiments we assume that we can walk from every station to every other station within 10 minutes walking radius. The *events graph* nodes are arrival and departure events of a station and are interconnected by four types of links: *departure links*, *continue links*, *changing links*, *waiting links*. Typically the *Time-Expanded* model has three types of nodes: *arrival node*, *departure node* and *transfer node*. Eliminating *transfer nodes* and all the links from *transfer nodes* to *departure nodes* in the typical *Time-Expanded* model and connecting *arrival* and *departure* nodes directly allowed us to reduce space requirements by 30%. In addition to the storage saving this modification also speeds up precomputation time. The described graph is illustrated in Figure 1.

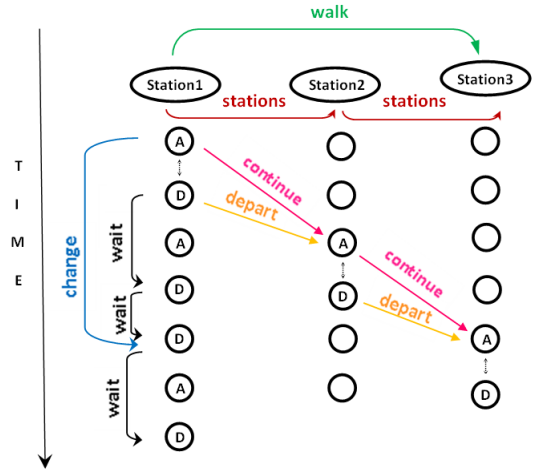

Figure 1. The two layered, time expanded graph. The first layer consist of three stations connected with *station links* and *walking links*. The second layer consist of arrival (A) and departure (D) events connected with *depart links*, *continue links*, *change links* and *wait links*.

Modeling the public transport in this way allows us to treat all different modes as a single mode.

3.2 Hub Nodes

Consider an example in Figure 2 below.

Assume there are three bus (or e.g. train) services: $A \to B \to C \to D \to F$, $H \to B \to C \to D \to E$ and $G \to D \to E$. We want to travel from A to E. Clearly, at some point we will have to change a service and it can be either at B, C or D. Now, there is no particular reason for us to change a service at C. On the other hand, potentially we may change a line at B or D. Therefore, we will identify B and D as *hub stations* and will refer to C as a *non hub station*. We observed that in reality (for Sydney and NSW public transportation networks), only relatively small portion of all stations, $15\% - 20\%$, are hubs. More formally, let $G_S = (V_S, E_S)$ to be a

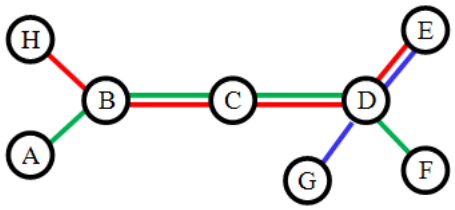

Figure 2. Example of hub and non-hub nodes

station graph, as described in Section 3.1. Denote S_i be a set of all services that depart from station $s_i \in V_S$. Then set of hub nodes is defined as

$$H = \{s_i \in V_S \mid \exists s_j.\ s.t.\ (s_i, s_j) \in E \ \wedge\ S_j \subset S_i\} \bigcup$$

$$\{s_j \in V_S \mid \exists s_i.\ s.t.\ (s_i, s_j) \in E \ \wedge\ S_i \subset S_j\} \bigcup$$

$$\{s_i, s_j \in V_S \mid (s_i, s_j) \in E \ \wedge\ S_j \neq S_i\}$$

3.2.1 *Pseudo code for identifying hub-nodes*

Using the notations from previous Section, we present a pseudo code for determining *hub nodes*.

Algorithm 1 Pseudo code for identifying hub-nodes

```
for (s_i ∈ V_S)
  for ((s_i, s_j) ∈ E)
    if (S_j ⊂ S_i)
        H := H \ s_j
        H := s_i
    else if (S_i ⊂ S_j)
        H := H \ s_i
        H := s_j
    else if (S_i ≠ S_j )
        H := s_i
        H := s_j
```

We notice that if we know the optimal route between any hub station to any other hub station, then a shortest route between any two nodes (not necessarily hubs) can be found immediately as described in the next Section.

3.3 Extracting the optimal path

We make an important observation that for every two "far away" non hub stations A and B, the following schema holds.

Generally, if a station s_i is not a hub node, there is only one succeeding station s_j, such that there is a service between s_i to s_j, i.e. $(s_i, s_j) \in E_S$. Otherwise s_i would be a hub node itself. Similar holds for S_j, etc... Therefore, referring to Figure 3, there is only one direct way to follow from any non hub node A to its first hub node H_A and from H_B to B. In other words, any service that departs from A will certainly arrive to H_A. Similarly any service that arrives to B will certainly depart from H_B (cause otherwise B would be a hub as well). It brings us to the following idea. If we only knew the shortest path from any hub node to any other hub node, it would

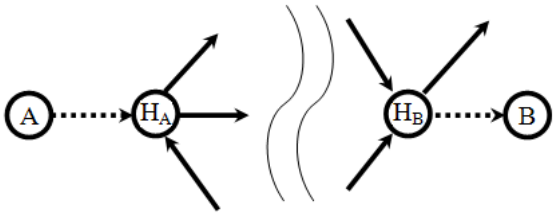

Figure 3. Example of query between two non hub stations

give us very fast, simple and intuitive algorithm for finding shortest path between any two stations.
Given a query between A and B:
(i) Start from A and follow to it's first (outgoing) hub node H_A
(ii) Traverse backwards from B find it's first (incoming) hub node H_B.
(iii) Fetch from precomputed database the optimal route $H_A \rightsquigarrow H_B$.
(iv) Combine all three segments together to obtain an optimal path .
Since eventually public transportation networks are inherently time dependent, we are interested in a query $A@t \rightarrow B$. Adding a time dimension to the algorithm above is relatively simple as well:
(i) Start with first service that departs at time $t_1 > t$ form A and follow until first hub node H_A arriving there at time t_2.
(ii) Fetch from the precomputed database the optimal route $H_A@t_2 \rightsquigarrow H_B$
(ii) Continue to B with a direct service that departs from H_B at time $t_3 > t_2$
(iv) Combine all three segments together to obtain an optimal path .

Although the obtained path, in theory will be the fastest, in practise it may not be very convenient and involve unnecessary waiting (for example if services departing from H_A are infrequent) and unnecessary change between services at the hub nodes. In order to address this we do some "after-analysis" of the obtained route: if in the obtained route there is a relatively long waiting (say more than 10 mins) at H_A, we check if we can leave A later and still eventually to arrive at the same time. We will discuss in a Section 4.6 in more details how to apply the obtained results into the real world.

4 TRANSIT with Hub Nodes

Assume for a moment that our world is a public transport network, i.e. we start and finish our journey at *event nodes*. We start with solving a single objective problem, e.g. we are interested to find the fastest way to get from station A to station B starting our journey at time t, i.e. $A@t \rightarrow B$. This problem is equivalent to finding an earliest arrival time, given the departure time. In reality the user may be interested in finding a route with the latest departure time, given an arrival time, which we will denote as $A \rightarrow B@t$. This can be answered efficiently by simply applying exactly the same algorithm but using backward links.
From Section 3.3 we know that it is enough to know the optimal route only between pair of hubs in order to easily reconstruct the optimal route between any two nodes. Therefore our aim will be to efficiently answer the queries between any two hubs. For this we will use the modified TRANSIT algorithm. Similarly to our previous algorithm this one also consists of two phases - precomputation and query.

4.1 Precomputation

In order to identify *transit nodes* we exploit the fact that the public transport stations have geographical coordinates associated with them.

4.1.1 Determining the Transit Nodes

This stage is performed on the *station graph* layer of our network. Consider a bounding box of the given network and the subdivision of this box into $g \times g$ cells. Let's C denote a cell. Intuitively, the transit node is an access node to (or from) a cell C. Let S_{inner} and S_{outer} be squares around cell C in the center, as depicted in Figure 4. In what follows, we will look for transit nodes near the border of S_{inner}, whilst S_{outer} will determine where long distance paths end. Generally, the size of the squares can be arbitrary without compromising correctness, but it directly impacts the preprocessing time, number of transit nodes, and consequently the storage space and query time. Experimentally we verified that the sizes of S_{inner} and S_{outer} of 3×3 and 5×5 cells produce a good compromise between precomputation time, storage space and finally average query time. We differentiate between two types of *transit* nodes - access nodes *into* the cell and *out* of the cell. Let V_C be a set of nodes as follows: for every link that has one of its endpoints inside C and the other outside C, if the link is bidirectional V_C will contain the endpoint inside C and for one directional link V_C will contain both of its endpoints. Similarly we define V_{inner} and V_{outer}, considering links that cross S_{inner} and S_{outer} accordingly. Now, the set of transit nodes for the cell C is the set of nodes $v \in V_{inner}$ with the property that there exists a shortest path from/to some node in V_C to/from some node in V_{outer} which passes through v. We associate every node inside C with the set of transit nodes of C. Next, we iterate over all cells and similarly identify transit nodes for every other cell.
Remark: It wouldn't be correct simply to "shortcut" between hub nodes and to determine transit nodes in the induced network. This is since in order to get from one hub to another, we still may need to change a service at the non hub node, therefore we can not eliminate them at this stage.

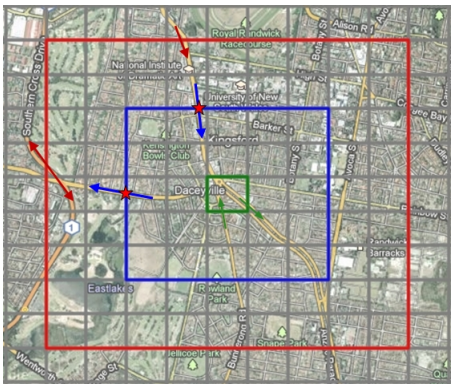

Figure 4. Example of the grid, cells, inner (blue) and outer (red) squares, different types of crossing links and two types of transit nodes

4.1.2 Computation and storage of Subpaths

This stage is performed on the *events graph* layer of our network. We precompute the following shortest routes and store them in three ta-

bles: (i) *hub-to-transit*: for every hub station H, from every departure event of H to every transit station node of H (ii) *transit-to-transit*: from every departure event of transit station node to every other transit station node and (iii) *transit-to-hub*: for every hub station H, from every departure event of its associated transit station node to H.

4.2 Local Search Radius

In order for TRANSIT algorithm to be correct, we need to guarantee that for every two nodes src and dst both of them (i) outside S_{outer} of each other and (ii) their corresponding S_inner squares do not overlap. Therefore, we define a local search radius to be the size of S_inner plus the distance from S_{inner} to S_{outer}. Two nodes for which horizontal or vertical distance is greater than local search radius are considered to be "far away" and a query between them called *a global query*. All other queries are *local*.

4.3 Query (time only)

Given a global query $A@t \rightarrow B$, between two hub nodes A and B, we find the fastest journey time as follows. We fetch transit station nodes of A and B, T_A and T_B accordingly. Let $\tau_A \in T_A$ and $\tau_B \in T_B$ be transit nodes of A and B. For every τ_A we fetch $c_1 = cost(A@t, \tau_A)$. Let's assume that this route arrived to τ_A at time t_1. Then we fetch $c_2 = cost(\tau_A@t_1, \tau_B)$. Finally, assuming that the fastest route from τ_A at time t_1 arrived to τ_B at time t_2, we fetch $c_3 = cost(\tau_B@t_2, B)$. Eventually the total travel time will be $c_1 + c_2 + c_3$. Iterating over all $\tau_A \in T_A$ the cost of the fastest route will be the one that yields minimal $c_1 + c_2 + c_3$. In case $A@t \rightarrow B$ is a local query, we just apply any efficient search algorithm, A* for example.

4.4 Query (actual path)

Unlike road networks, where for many applications returning the time (or distance) is good enough, in public transport networks users will be usually interested in concrete trip directions. More precisely, besides knowing on which service to board at the start of their journey, we need to provide instructions where they should change services. Luckily, the vast majority of our journeys consist of very small number of transfers, 2-3, 4 and more on very rare cases. Observing this, during precomputation phase for every precomputed pair we are storing the full instructions at what station and at what time to depart with the next services. Alternatively we can store only the first transfer instruction and reconstruct the whole journey by iteratively applying the Query from the next transfer station. Again the number of iterations should be very small.

4.5 Dealing with Multiobjectiveness

In addition to find the fastest connection between two points, the user usually consider also other criteria, such as the cost of tickets, hassle of interchanging between services, etc. Moreover, different users can have different preferences over these criteria. For example a businessmen may try to optimize his travel time, a student may try to minimize his costs and a visitor may wish to avoid changing services in order not to get lost. There are several ways to deal with multiobjective optimization [9], in this work we choose the normalization approach, by introducing a linear utility function as in [9]. This approach reduces the multi-criteria problem to a single-criterion optimization, which we can solve as described. We will precompute the tables for different values of the linear coefficients of such a linear combination. For example a businessmen will try to optimize his travel time, a student will try to minimize his costs and a visitor will try to avoid interchanging services in order not to get lost.

4.6 Providing multiple results in the real world

In the real world, we may need to walk from our home to our first public transport station and from the last station to our final destination. Also, in the real world, most of the time we may not be interested in the theoretically optimal result, but in more practical one. For example, one may prefer a journey that is only 2 mins longer to a shorter journey that involves an additional change. Moreover, in real life we would prefer to wait a little bit now if we know that eventually we may arrive earlier, for example to wait for an express bus. In addition we all like choices, therefore the system will be more user friendly if it could provide multiple attractive alternatives to the user. In order to cover those real life scenarios and provide the user several attractive suggestions we will run our Query starting from different stations around the user's starting location and different times around his specified departure/arrival time t. Also, at step (ii) we will fetch several routes from H_A to H_B. From the all combined routes, we will choose, say the best five. Experimentally we could see that this heuristics provide us with the most reasonable results that a person would make.
Another important aspect we consider is robustness of the provided solutions. In practice, public transport often runs late due to traffic congestion or accidents or other unpredicted events. Missing a connection by one minute may cost us an hour in our total journey time, if the next connection is infrequent and departs say only once an hour. In order to minimize such occurrences and to make the system more reliable and user friendly we identify those trips and warn the user about risky connections. Another option is to ask the user to define his risk adverseness and/or preferred time for connection between two different services. Then the user will choose his preferred trip according to his risk adverseness.
In the next Section we present a speed up technique to efficiently precompute and store the optimal routes between hubs.

4.7 Speed up

Since we don't precompute shortest paths between any two stations, but only between, say 15% of the stations, at the first glance the proposed algorithm may seem appealing. But considering the fact that actually we are precomputing an optimal route for every event of the hub station, precomputation time and memory requirements are still relatively large. Looking closely, we can observe particular patterns for every service during the day. Luckily number of these patterns is not large, for example it may take for some service 20min. at night, 30min. in the morning, 25mins during the day and 35min. at evening. Essentially this is the same service, but it may be affected and take different time due to city traffic patterns. As a preprocessing stage, we identify those patterns and then at stage (ii) we precompute only for the patterns rather than for every departure event. For example, if there is a service from A to B, departing say every 10 mins and it stops at the same stops at exactly the same times there is no need to precompute for the same pattern several times. This observation allowed us to achieve a significant reduction in both preprocessing time, and storage space.

5 Implementation and Experiments

The proposed algorithm was implemented using the Java programming language and the experiments were performed on PowerEdge 1950 server, with those specifications: CPU: 2 x 2.00GHz Intel Quad Core Xeon E5405, Memory: 16 GB. The presented results were obtained using the Sydney metropolitan and state of NSW public transport multimodal network consisting of buses, trains, ferries, lightrail and monorail modes. The graph representing Sydney network contains $9.3K$ station nodes, $2.1M$ event nodes and $8.1M$ links. The graph representing the whole NSW network contains $46.3K$ station nodes, $6.7M$ event nodes and $23.2M$ links.
Given a grid size $g \times g$ and sizes of S_{inner} and S_{outer}, for every query it is very easy to verify whether it is a global or a local. We just need to check if the two nodes within local radius of each other. Therefore by simple sampling of many random queries we can have a good estimate of the percentage of global vs.local queries. It allows us to fine tune the sizes of grid, S_{inner} and S_{outer} to achieve the desirable percentage of global queries. Larger values for S_{inner} and S_{outer} normally yield smaller number of transit nodes, consequently requiring smaller memory requirement, but also covers a smaller number of global queries. We experimented with different sizes of the grid, S_{inner} and S_{outer} and below present results demonstrating tradeoff between percentage of global queries and memory/time requirements of the offline stage.

Area	Global queries	Grid size (cells)	$\|T\|$	Storage Mb.	Precomp. time (hours)
Sydney	70%	70 x 70	1174	321	0.7
Sydney	80%	89 x 89	1505	446	0.9
Sydney	90%	136 x 136	2256	825	1.2
NSW	70%	42 x 42	1023	215	1.5
NSW	80%	59 x 59	1733	375	2.5
NSW	90%	94 x 94	3270	1200	6.5

Table 1: Experimental results for Sydney and NSW public transport network using different grid sizes, with $S_{inner} = 3 \times 3$ and $S_{outer} = 5 \times 5$ squares comparing between $|T|$ number of transit nodes, storage space and precomputation time.

Applying TRANSIT algorithm in a naive way, without including *hub nodes* and any speed ups yields memory requirements up to 45Gb for Sydney and 50Gb for NSW networks with the estimated precomputation time of several days.

Below is a table summarizing the average query time for a single query of different types, specifically: hub-to-hub (HH), station-to-station (SS), location-to-location (LL).

Area	Global queries	Grid size (cells)	HH	SS	LL
Sydney	70%	62 x 62	91	95	104
Sydney	80%	80 x 80	70	76	85
Sydney	90%	121 x 121	49	55	65
NSW	70%	38 x 38	70	76	85
NSW	80%	55 x 55	89	95	105
NSW	90%	87 x 87	66	71	80

Table 1: The average query time (in micro seconds) for a single query of different types: hub-to-hub (HH), station-to-station (SS), location-to-location (LL).

6 Conclusions and Future work

In this work we presented a novel approach for finding optimal connections in public transportation network. We presented an experimental results using the Sydney metropolitan and state of New South Wales public transportation network. From the results we believe that the idea has a great potential and plan to improve it further. Our future work includes improving the preprocessing time and reducing the database tables storage space, if possible whilst preserving optimality. There are several promising directions we are interested to investigate in order to achieve this, such as using an adaptive grid rather than simple, fixed grid or partitioning the underlying graph into clusters in a completely different manner. Hub nodes, the way we defined it, can be seen as a second level of hierarchy over the station graph, therefore it looks natural to combine this idea with Contraction Hierarchies [6] that works very well for road networks. In addition, we are interested to provide more rich set of alternatives to the user. Also we would like to extend this idea to fully realistic inter modal journey planer which includes combination of private transport (e.g. car, motorbike, bicycle) and public transport and incorporates real time updates for both traffic conditions and public transport actual location and estimated time of arrival.

ACKNOWLEDGEMENTS

The first author would like to thank Daniel Harabor for helpful discussions.

REFERENCES

[1] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck, 'Highway dimension, shortest paths, and provably efficient algorithms', in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pp. 782–793, Philadelphia, PA, USA, (2010). Society for Industrial and Applied Mathematics.
[2] Hannah Bast, 'Efficient algorithms', chapter Car or Public Transport–Two Worlds, 355–367, Springer-Verlag, Berlin, Heidelberg, (2009).
[3] Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger, 'Fast routing in very large public transportation networks using transfer patterns', in *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'10, pp. 290–301, Berlin, Heidelberg, (2010). Springer-Verlag.
[4] Holger Bast, Stefan Funke, and Domagoj Matijevic, 'Transit ultrafast shortest-path queries with linear-time preprocessing', in *9th DIMACS Implementation Challenge*, (2006).
[5] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes, 'In transit to constant time shortest-path queries in road networks', *Proc. 9th Workshop on Algorithm Engineering and Experimentation (ALENEX)*, (2007).
[6] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling, 'Contraction hierarchies: Faster and simpler hierarchical routing in road networks', in *Workshop on Experimental and Efficient Algorithms*, pp. 319–333.
[7] Transit Google, *http://www.google.com/intl/en/landing/transit*.
[8] Toby Walsh Leonid Antsfeld, 'Finding multi-criteria optimal paths in multi-modal public transportation networks using the transit algorithm', *19th world Congress on Intelligent Transport Systems, Vienna*, (2012).
[9] Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis, 'Timetable information: models and algorithms', in *Proceedings of the 4th international Dagstuhl, ATMOS conference on Algorithmic approaches for transportation modeling, optimization, and systems*, ATMOS'04, pp. 67–90, Berlin, Heidelberg, (2007). Springer-Verlag.
[10] TransportInfo NSW, *http://www.131500.com.au/*.
[11] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis, 'Experimental comparison of shortest path approaches for timetable information', in *Algorithm Engineering and Experimentation*, pp. 88–99, (2004).
[12] Peter Sanders and Dominik Schultes, 'Robust, almost constant time shortest-path queries on road networks', in *IN: 9TH DIMACS IMPLEMENTATION CHALLENGE*, (2006).

Improved Agent Based Algorithm for Vehicle Routing Problem with Time Windows using Efficient Search Diversification and Pruning Strategy

Petr Kalina[1] and **Jiří Vokřínek**[2]

Abstract.

We suggest an improved algorithm for the vehicle routing problem with time windows (VRPTW). The algorithm is based on negotiation between a fleet of agents corresponding to the routed vehicles using a set of generic negotiation methods and a state-of-the-art insertion heuristic. A search diversification and pruning strategy is introduced which allows for a wide range of competing algorithm instances to be instantiated and efficiently managed throughout the solving process. Experimental results on the widely used Solomon's and the extended Homberger's benchmarks prove that the algorithm is broadly competitive with respect to the established centralized state-of-the-art algorithms equalling the best known solutions in 64% of the cases with an overall relative error of 2.4%, thus achieving a new best known result for an agent based algorithm to date. The vastly improved negotiation process and the inherent parallelization features provide for excellent anytime features, outperforming even the state-of-the-art algorithms in this respect.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is one of the most widely studied problems in the the domain of logistics. The VRPTW is a problem of finding a set of routes from a single depot to serve customers at geographically scattered locations. Each customer is visited by exactly one route with each route starting and ending at the depot. There are two constraining factors that need to be considered: (i) for each route the sum of demands of the customers served by the route must not exceed the capacity of the vehicle serving the route (capacity constraint) and (ii) the service at each customer must begin within a given time interval (time window constraints). The primary objective of the VRPTW is to find the minimum number of routes servicing all customers.

Recent years have seen a growing interest of the scientific community in multi-agent systems as an emerging choice for modeling complex systems with highly dynamic, heterogenous, potentially non-cooperative or privacy conscious environments. The real world applications of routing algorithms often display many of the above mentioned characteristics. Traditionally, the majority of VRPTW related research is concerned with centralized algorithms, with the agent based studies being scarce and typically concerned rather with the real-world applicability of the presented algorithms than their thorough performance and theoretical analysis.

Several very recent agent based works [6, 5] reported a competitive solution quality and provided a sound theoretical analysis of the respective algorithms. Within this paper we extend these works by introducing an improved parallel algorithm based on agent negotiation (Section 3) using an efficient search diversification and pruning strategy (Section 4). The performance, convergence and other aspects of the algorithm are assessed using the widely used benchmark sets of Solomon [14] and Homberger [4] (Section 5).

2 Related Work

We refer the reader to a comprehensive survey of VRPTW algorithms up to year 2005 provided by [1]. Also, the list the contemporary leading algorithms with respect to the benchmarks used within this study is maintained at [13].

In [9] the authors present an algorithm based on the *ejection pool* principle. The algorithm is based on performing very good unfeasible insertions of customers to individual routes, followed by an ejection procedure in which the feasibility is recovered by ejecting some other customers from the unfeasible routes. The algorithm equals the best known cumulative number of vehicles (CVN) of 405 on the Solomon's instances.

An improved algorithm presented in [11] further employs a specific local search strategy guiding the ejections. Also, a feasible insertion mechanism denoted as *squeeze* as well as a search diversification *perturb* procedure are employed throughout the solving process boosting the algorithm's convergence. The algorithm provides for the contemporary best known CVN of 10290 over the extended Homberger's benchmark set.

An agent based algorithm for VRPTW and PDPTW is presented in [6] based on the Task Agent — Allocation Agent — Vehicle Agent hierarchy. The tasks are allocated to a fleet of the Vehicle Agents in a series of auction steps based on the well known contract net protocol (CNP). A set of improvement methods is introduced based on agent negotiation, that can be executed either *dynamically* after each auction step or *finally* after all tasks have been allocated. Several initial task orderings are introduced, providing for a number of alternative *particular solvers*. These solvers are then run in parallel with the best result being returned. The algorithm provides for a CVN of 10889 over the Homberger's extended benchmark set. However, as already shown in [14] and [5], a cost structure targeting directly the temporal aspects of the problem has been proved superior to the used travel time savings heuristic. Also the set of competing particular solvers

[1] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, peta.kalina@gmail.com
[2] Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague, jiri.vokrinek@fel.cvut.cz

is static accross all instances, resulting in a significant performance overhead.

The algorithm presented in [5] is based on similar concepts. However, it introduces a cost structure corresponding to the *slackness savings heuristic* similar to [10] as well as a set of more refined improvement methods. The algorithm is run using three initial task orderings with the best result being considered. The algorithm provides for a contemporary best known results achieved by an agent based algorithm for both the Solomon's and the extended Homerger's benchmarks with a respective CVN of 429 and 10609. However, the traversed search space is quite narrow due to the static choice of orderings and algorithm configurations. The above mentioned performance overhead is not addressed as well.

To our knowledge, the only other agent based algorithm with results comparable to those presented within this work is presented by [8] achieving a CVN of 436 over the Solomon's benchmark set. The remaining relevant studies [3, 7, 2] focus on real-world derived scenarios using ad-hoc problem sets and therefore do not provide comparable performance information.

3 Negotiation Based Allocation Algorithm

The main contribution of this work is (i) the extension of concepts presented in [5] and [6] by introducing an improved parallel algorithm based on agent negotiation with an efficient search diversification and pruning strategy and (ii) the assessment of its relevance with respect to the state-of-the-art centralized as well as previously presented agent based algorithms.

3.1 Core Agent Framework

The algorithm is based on a three layer basic architecture appearing also in [5] featuring a top layer represented by a Task Agent, middle layer represented by an Allocation Agent and a fleet of Vehicle Agents present at the bottom level of the architecture.

Task Agent acts as an interface between the algorithm's computational core and the surrounding infrastructure. It is responsible for registering the tasks and submitting them to the underlying Allocation Agent.

Allocation Agent instruments the actual solving process by negotiating with the Vehicle Agents. The negotiation is conducted based upon task commitment and decommitment cost estimates provided by the Vehicle Agents.

Vehicle Agent represents an individual vehicle serving a route. It provides the Allocation Agent with the above mentioned inputs. These are computed based on local (private) Vehicle Agent's plan processing.

Characteristic to the agent decomposition is the clear distinction between Vehicle Agents' local planning and the global planning managed by the Allocation Agent. This allows for a transparent inclusion of the typical real-world concepts such as loading constraints, heterogenous cost structures for individual vehicles or a more complex commitment semantics.

3.2 Improved Agent Negotiation Process

At the core of the algorithm is the negotiation carried out between the Allocation Agent and the fleet of Vehicle Agents. The improved negotiation process is illustrated by Figure 1.

```
Input : Set of problem instance tasks T
Output : Improving sequence of solutions

Procedure negotiate(T)
begin
 1:  σ_best := null;
 2:  foreach (setting S ∈ diversified algorithm settings set Δ)
 3:   foreach parallel (ordering O ∈ diversified orderings set Ω)
 4:    vn := (σ_best = null ? CLB(T) : vn(σ_best) − 1)
 5:    repeat
 6:      T_O := apply O on T;
 7:      σ_O := initial fleet of vn empty vehicles
 8:      foreach task t ∈ T_O
 9:        v := auction(t, σ_O);
10:        σ_O := commit(t, v);
11:        σ_O := dynamicImprove(S, σ_O, T_O);
12:      end foreach
13:      σ_O := finalImprove(S, σ_O, T_O);
14:      if (feasible(σ_O))
15:        if (σ_best = null or vn(σ) < vn(σ_best))
16:          σ_best := σ_O;
17:          output(σ_O);
18:        end if
19:        vn := vn(σ_best) − 1;
20:      else
21:        if (σ_best = null or vn(σ) < vn(σ_best) − 1)
22:          vn := vn + 1;
23:        else
24:          store σ_O for orderings pruning
25:          break repeat;
26:        end if
27:      end if
28:    end repeat
29:   end foreach parallel
30:   prune(Ω);
31:  end foreach
end
```

Figure 1. Improved agent negotiation process

The process begins with resetting the temporarily best found solution σ_{best} (line 1). Follows the selection of a particular algorithm configuration from a set of increasingly complex algorithm configurations Δ (line 2). A particular algorithm configuration $C \in \Delta$ is given by specifying the particular improvement methods to be used for the *dynamic* and *final* improvement steps at lines 11 and 13. There are three improvement methods defined (e.g. the $ReallocateAll$) that are refinements of the similar methods presented in [5] providing for an efficient search diversification on the algorithm configuration level, described in detail in Section 4.1.

Follows the selection of the particular instance task processing ordering (line 3). Based on previous findings [5] the agent based negotiation mechanism provides a very good convergence given a fitting task ordering is provided. The set of orderings Ω is obtained by two specific ordering diversification operators described in detail in Section 4.2 applied to a set of canonical analytically sound orderings, providing for the search diversification on the task ordering level.

Lines 3–29 outline the internal task allocation loop, denoted also

as a particular *algorithm instance* parameterized by $C \in \Delta$ and $O \in \Omega$. All algorithm instances corresponding to the currently processed algorithm configuration $C \in \Delta$ are processed in parallel, providing for a set of solutions $\sigma_O, O \in \Omega$. After these algorithm instances have finished, the results are used to prune the set Ω in an effort to effectively direct the search in the most feasible direction.

Thus for each algorithm instance an initial empty partial solution σ_O is instantiated, corresponding to a fleet of empty vehicles (line 7). Individual tasks are allocated to σ_O in a series auction steps (lines 9,10), followed by a particular dynamic improvement step (line 11), with the final improvement step being applied after all tasks have been processed (line 13). As the task allocation process is vastly similar to the one presented in [5], for the sake of clarity we present only a simplified version, neglecting some of the finer details not relevant for the contribution presented within this work.

In a departure from the previous works [6, 5], the initial size of the fleet (number of vehicles - VN) is always determined to target a new best found solution with respect to the current best solution σ_{best}. In both above mentioned works the individual algorithm instances were initialized with a VN corresponding to the theoretical lower bound count on the number of vehicles (LBVN). In case the resulting solution was not feasible (e.g. contained uncovered customers), the instance was restarted with a VN incremented by 1 until a feasible solution was found. There are several drawbacks to this strategy.

The discrepancy between the LBVN and the VN eventually found by a particular algorithm instance has a multiplicative effect on the complexity of the resulting negotiation process due to the above mentioned restarts strategy. Considering an instance using an unfitting ordering a higher number of restarts is necessary before a feasible solution is found. Even more intriguingly, such a solution is likely to be superseded by a solution provided by an algorithm instance with a more fitting ordering. Thus the majority of the processing time is actually spent by constructing solutions that will be most likely discarded and thus wasted.

Also, by starting from the LBVN, *all* competing algorithm instances are bound to be restarted at least a fixed number of times corresponding to the difference between the VN of the best eventually found solution and the LBVN. Thus the complexity of the resulting algorithm depends heavily on the nature of the instance being solved, as the interdigitation of the underlying multiple bin packing problem and the time windows constrains may require a much greater VN than the LBVN. Lastly, the determination of the LBVN is actually a NP-hard problem in itself [10]. The algorithms presented in [6] and [5] use therefore a greedy $O(N^3)$ approximation, requiring further processing time.

Thus, as already mentioned, an alternative restart strategy is introduced, with the initial VN being set to improve on the best found solution to date σ_{best}. In case such a solution is not available (e.g. during the initial stages of the solving process), an alternative capacity based LBVN (CLB) is used, computed based on the vehicle capacity and the cumulative demand of all customers [6] that can be computed in $O(1)$ time (line 4).

After the task allocation has finished (line 14), depending on the feasibility of the resulting solution σ_O and the current σ_{best} (which may differ from the initial value from the beginning of the instance run), the instance can be (i) pruned (line 25), (ii) restarted with an increased initial VN (line 22) or (iii) restarted with a new best found targeting VN (line 19).

The first situation occurs when σ is unfeasible with a VN higher than the contemporary $vn(\sigma_{best}) - 1$. In such a case a restart with an increased fleet size $vn + 1$ cannot potentially yield a result superior to σ_{best} and is therefore redundant. In case the current σ_{best} has a higher VN or has not yet been set, the restart is feasible, corresponding to the second above mentioned situation. On the other hand when a feasible solution σ_O is found the process can be restarted with a VN targeting a new best found solution. In case σ also represents the new contemporary best found result, it is stored and output (lines 16, 17).

After all algorithm instances have finished and before a new algorithm configuration $C \in \Delta$ is processed, the set of orderings Ω is pruned for the next phase, based upon the solutions $\sigma_O, O \in \Omega$ stored during the previous phase (line 30). Two alternative ordering pruning strategies are introduced in Section 4.2.2.

Thus the search diversification is achieved by using diversified sets Δ and Ω with the resulting performance penalty being partially offset by introducing an *overall pruning strategy* consisting of (i) an ordering pruning strategy limiting the number of algorithm instances run for the more complex algorithm configurations and (ii) a strategy allowing for an efficient management of the overall solving process based on reusing the already achieved results to effectively limit (prune) the number of individual algorithm instance restarts.

4 Search Diversification and Pruning Strategy

As mentioned above, the broader search space coverage of the presented algorithm is achieved by the (i) diversification on the algorithm configuration level by means of fine grained control over the improvement methods and (ii) diversification on the instance task ordering level by means of two specific ordering diversification operators.

4.1 Algorithm Configuration Diversification

The three improvement methods used for the dynamic and final improvement steps are the *ReallocateAll* method, the ε*-ReallocateWorst* method and the ε*-ReallocateRandom* method. Their basic semantics is carried over from [5]. The methods are based on iterating through portions of each agent's plan and *reallocating* corresponding customers in a series of auction steps to agents with a better commitment cost estimates, thus improving the partial solution σ_O under construction. The methods differ by the specific order in which the customers in an individual agent's plan are processed. For purposes of this work these methods were further improved by introducing the *loop count* parameter, affecting the number of repetitions of the above mentioned process on a single agent's plan. This allows for a configurable tradeoff between the method's complexity and its relative strength.

The algorithm configurations basic notation is carried over from [5], with *B* (*Basic*) denoting a configuration with neither improvement step applied, *FI* (*Final Improvement*) denoting a configuration with only a $ReallocateAll$ based final improvement step being applied and *DI* denoting configurations extending the *FI* configuration with a particular dynamic improvement step being applied as well.

The set Δ thus comprises of a *B* configuration, a *FI* configuration and three subsets of *DI* configurations each corresponding to a particular method being used for the dynamic improvement step. These subsets are further diversified by using an increasing *loop count* parameter setting.

4.2 Ordering Diversification and Pruning

As mentioned above, the initial set of orderings Ω is generated from the set of analytically sound orderings using two specific operators.

Table 1. Performance of the algorithm compared to the best known results — CVN and relative error

Type	SotA [9, 11]	Agents [8]	Agents [6]	Agents [5]	*BP*	*CSP*	*BP+CSP*	$\Delta \times \Omega$
All	10695	–	–	+343 (3.2%)	+290 (2.7%)	+258 (2.4%)	+254 (2.4%)	–
100	405	+31 (7.7%)	–	+24 (5.9%)	+17 (4.2%)	+16 (4.0%)	+16 (4.0%)	–
200	694	–	–	+21 (3.0%)	+18 (2.6%)	+12 (1.7%)	+12 (1.7%)	–
400	1380	–	–	+38 (2.8%)	+35 (2.6%)	+31 (2.2%)	+29 (2.1%)	+29 (2.1%)
600	2065	–	–	+56 (2.7%)	+51 (2.5%)	+43 (2.0%)	+43 (2.1%)	+41 (2.0%)
800	2734	–	–	+89 (3.3%)	+76 (2.8%)	+72 (2.6%)	+71 (2.6%)	+70 (2.6%)
1000	3417	–	–	+115 (3.4%)	+92 (2.7%)	+84 (2.5%)	+83 (2.4%)	+82 (2.4%)
200–1000	10290	–	+609 (3.1%)	+319 (3.1%)	+273 (2.7%)	+242 (2.4%)	+238 (2.3%)	–

The canonical set contains the three orderings introduced in [5] with two additional orderings — the *larliest expiry first* (LEF) based on the *latest service start* value of individual tasks' time windows and the *Most Distant First* (MDiF) based on the distance of individual customers from the depot.

4.2.1 Ordering diversification operators

The two presented operators were introduced to provide means of *diversification* by providing for a diversified set Ω of initial instance task orderings. The *k-perturb(O)* operator is based on randomizing the order of sub-sequences of length k on the underlying set of tasks ordered by O. The *k-mixin(O_1, O_2)* operator combines two orderings by applying the secondary ordering O_2 to sequences of k tasks on the underlying set of tasks ordered by O_1.

As opposed to the well known *ordering crossover* and *mutation* operators used by the genetic ordering based algorithms [12], these two operators were specifically tailored to allow for traversing a neighborhood that is very close to the original analytically sound orderings and thus preserve the *nature* of these orderings. This effort was motivated by the previous findings [5], as it was proved that the analytically sound orderings significantly outperform randomly generated orderings. However, the rigorous assessment of suitability of known crossover operators or eventual applicability of a genetic based approach to identify the most fitting ordering is not part of this study and therefore remains an interesting area of future research.

4.2.2 Ordering pruning strategies

As outlined by the main negotiation process, after a particular algorithm configuration has been processed, the set of orderings is pruned based on the achieved results. Due to the fact that some results correspond to unfeasible partial solutions, instead of using the VN as the indicator of the resulting quality, we introduced the *average route size* metrics, corresponding to the average number of customers in a single route, providing for an ordering of the set of (partial) solutions σ_O. Two fundamentally different pruning strategies were introduced.

The *Basic Pruning Strategy* (*BP*) is based on the assumption that the optimal ordering may differ significantly for each problem instance. Thus a given number of the best (with longest average routes) orderings to be kept for each processed algorithm configuration is specified and the set Ω can be effectively pruned based on the results from the faster configurations in an effort to provide the most complex configurations with the only best fitting orderings.

An alternative strategy denoted as the *Minimal Covering Set Pruning Strategy* (*CSP*) is based on the opposite assumption — that the set of orderings and their interesting neighborhoods is limited and partially static across all problem instances. Thus using only a specified subset of problem instances (in our case the 100 and 200 customer instances from the Solomon's and Homberger's benchmark set) the whole space of orderings and algorithm configurations $\Delta \times \Omega$ is traversed. A minimal covering set of orderings Ω_C is identified. These orderings are then used across the whole solving process. Obviously the success of such a strategy is based on the fact that the processed instances are similar in their nature as the training set, however we argue that real world scenarios often display such a characteristic and therefore such an approach is an interesting alternative way of pruning the set of orderings.

A hybrid strategy denoted as *CSP+BP* combines the two by applying the *BP* pruning while treating the orderings $O \in \Omega_C$ as *unprunable*.

5 Experimental Evaluation

The experiments were carried out using the established Solomon's and the extended Homberger's benchmark sets [14, 4], sharing the same basic attributes. Thus the complete set of 356 instances contains instances of 6 different sizes with 100, 200, 400, 600, 800 and 1000 customers respectively, with 60 instances in each set (except the Solomon's with 56 instances). For each set there are 6 instance types provided — the R1, R2, RC1, RC2, C1, and C2 type, each with a slightly different topology and time windows properties [14]. The inclusion of the extended Homberger's benchmarks provides for a relevant comparison with the established state-of-the-art centralized solvers that has been missing from most previous agent-based studies.

The algorithm prototype is implemented in JAVA programming language. The experiments were carried out using a 8G RAM AMD Athlon 2G Gentoo system running the 64-bit Sun JRE 1.6.0_22.

Four overall configurations were considered denoted as *BP*, *CSP*, *CSP+BP* and $\Delta \times \Omega$ corresponding respectively to the three previously defined pruning strategies being applied and a configuration with no ordering pruning employed thus traversing the whole diversified search space.

5.1 Overall Quality Analysis

The results summarizing the overall achieved solution quality are presented by Table 1. The *SotA [9, 11]*, *Agents [8]*, *Agents [6]* and the *Agents [5]* columns correspond to the state-of-the-art results for established centralized [9, 11] and the three previously presented comparable agent based algorithms [8, 6, 5] respectively. The remaining columns correspond to the four respective algorithm overall configurations. The presented results correspond the the *cumulative number of vehicles* (CVN) for the respective problem instance subsets, or the respective absolute and relative errors.

In overall the algorithm in it's full fledged *CSP+BP* configuration achieved a CVN of 10949 over all the benchmark instances, corresponding to a 2.4% relative error in terms of the primary optimization criteria with respect to the state-of-the-art centralized algorithms, equalling the best known results for 64% of the problem instances. With respect to previous agent based approaches this represents a new best known overall result, with the new best known result being achieved for 81 instances (23% of the cases).

The results thus strongly suggest that the presented diversification strategy is successful in terms of enabling the algorithm to traverse interesting areas of the search space resulting in a significantly improved solution quality.

5.2 Orderings Analysis

The experimental results presented within the Table 1 outline the respective success of the three alternative pruning strategies presented and the baseline $\Delta \times \Omega$ strategy without ordering pruning. The used set Ω corresponds to the 5 canonical orderings, extended by their *3-perturb* and *6-perturb* mutations and by their *10-mixin* and *20-mixin* combinations, providing for a set Ω of 65 instance task orderings.

The Ω_C corresponds to the minimal covering set of winning orderings from the $\Delta \times \Omega$ strategy being run over the Solomon's 100 and Homberger's 200 customer instances. Thus Ω_C contains 10 orderings, including the LEF canonical ordering attributing for the majority of wins, 4 orderings based on the *k-perturb* operator and 5 *k-mixin* based orderings.

The pruning strategy was set to retain 20 orderings based on the results from the first two *B* and *FI* algorithm configurations followed by a *minimal DI* configuration with the $ReallocateAll$ dynamic improvement method and a *loop count* $= 1$, upon which the set of orderings was pruned to mere two members processed by the rest of the algorithm configurations, corresponding to three subsets of configurations based on the three individual improvement methods being used for the dynamic improvement step with the *loop count* parameter being set to 3 and 6 respectively.

5.2.1 Diversification Operators

In overall the results indicate that the search diversification provided by the two introduced operators applied on the extended set of analytically sound orderings significantly improves the solution quality achieved by the solver. An additional experiment was carried out to determine a fitting parametrization of the mentioned ordering diversification operators, consisting of analyzing the corresponding Ω_C for the unrestricted $\Delta \times \Omega$ setting using a wider set of orderings generated by setting the k parameter to $k = 2, 4, 8, 16, 32, 64$ for both operators.

In case of the *k-mixin* operator the setting of the k parameter does not affect the success of the resulting ordering greatly, with the overall success roughly corresponding to the success of the two underlying orderings. However, with the *k-perturb* operator the lower k values dominate the higher values. These results suggest that (i) the *k-mixin* operator preserves the analytically derived soundness of the orderings irrespective of the k parameter settings providing for a flexible search diversification operation while (ii) the *k-perturb* operator diverges from the feasible analytically sound orderings with increasing k.

In terms of overall success the *k-mixin* operator slightly outperforms the *k-perturb* operator, however their simultaneous appearance in the identified set Ω_C mentioned in the previous section suggests that both provide for an effective alternative means of search diversification, given a fitting k is used for the *k-perturb* operator.

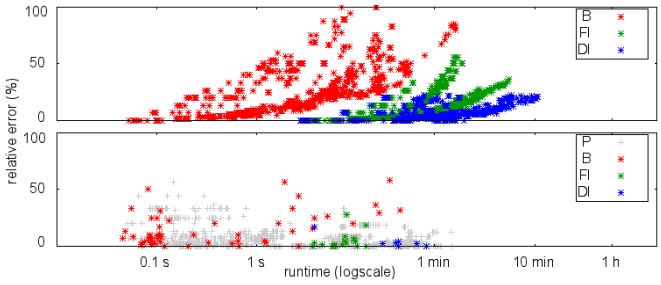

Figure 2. Individual algorithm instance results and runtimes with (bottom) and without (top) overall pruning strategy

5.3 Pruning Strategies

The results presented in Table 1 outline the success of the individual pruning strategies.

The *BP* strategy posted results that are clearly an improvement over the previous results based on a similar allocation process. With only the two best orderings being processed by the full blown solvers in the latter stages of the process the results suggest that the ordering pruning based on results of increasingly more complex algorithm configurations is quite successful. However, the convergence to the optimal ordering is not straightforward and — as outlined by the significantly superior results posted by the *CSP* strategy — often fails to identify the optimal ordering from the set Ω. We argue that this is due to two factors: (i) the metrics of a particular ordering success based on the average route size of the underlying (partial) solution is a very loose one, with many orderings achieving identical results and (ii) the relationship between relative success of two different algorithm configurations using the same instance task ordering is not straightforward, as the dynamic improvement methods process the tasks in the order as they appear in the corresponding agent's plan and thus the two solving processes based on an identical initial ordering can diverge significantly.

The very good results achieved by the *CSP* strategy suggests, that this strategy is actually very successful in identifying particularly good orderings or — in case of orderings produced by a particular ordering diversification operation — suggesting valuable neighborhoods of particular orderings. Considering the relatively small overall difference in achieved quality between the *CSP*, *CSP+BP* and the full $\Delta \times \Omega$ strategies the results suggest that the set of dominant orderings is relatively small and consistent over the whole benchmark set, further supporting the soundness of the *CSP* pruning strategy.

5.4 Runtime and Convergence analysis

The results presented within this study are based on an extended search based on diversified sets Δ and Ω inherently increasing the complexity of the overall process. To offset the increased complexity an *overall pruning strategy* is introduced based on ordering pruning and the improved negotiation process.

Figure 2 illustrates the improvements in runtime achieved by the introduced overall pruning strategy, corresponding to a subset of 16 problem instances from the 1000 customers instance set. The two

Table 2. Comparison with the state-of-the-art solvers in terms of runtime and convergence

	Nagata [11]	Lim [9]	*CSP+BP*	
Size	Avg. RT	Avg. RT	Avg. RT	Anytime RT
200	1 min	10 min	10 s	57 ms
400	1 min	20 min	2 min	300 ms
600	1 min	30 min	8 min	2 s
800	1 min	40 min	24 min	7 s
1000	1 min	50 min	54 min	14 s

compared settings correspond to (i) the *CSP+BP* strategy using the improved agent negotiation process outlined by Figure 1 (bottom) and (ii) the $\Delta \times \Omega$ strategy using a process based on [5] thus not employing any of the presented improvements (top). Individual points in the respective graphs correspond to the relative errors and runtimes recorded for individual algorithm instances. The results are grouped based on the underlying overall algorithm configuration with the pruned results being denoted as *P*.

The results suggest that the pruning strategies have a dramatic positive impact on the resulting runtime features of the algorithm. While in case of the original algorithm the biggest runtime penalty was implied by the least successful algorithm instances requiring numerous restarts with an incremented initial VN before a feasible solution is found, these instances are effectively pruned after only one restart in case of the presented algorithm. Furthermore by pruning the set of orderings the number of instantiated algorithm instances drops significantly with their increasing complexity with minimal impact on the resulting solution quality, providing for yet another massive boost in the algorithm's efficiency. Thus the recorded average composite single threaded time per processed instance (the sum of runtimes for all algorithm instances) is 41 minutes with the overall pruning in place, representing a massive 6.3 times improvement over the configuration not using the pruning strategy with 258 min.

The comparison in terms of runtime of the full fledged *CSP+BP* setting with the state-of-the-art algorithms is presented by Table 2. The listed values correspond to the average composite single threaded runtime. To provide illustration of the algorithm's anytime features the parallel runtime before the best solution is found is listed as well. The results confirm exceptional anytime features of the algorithm when its inherent parallelization features are fully exploited with the time before the best solution is found significantly outperforming all previous solvers. The composite runtimes are also competitive, especially considering that the underlying prototypal implementation is far from being performance optimized. We must note, however, that: (i) compared algorithms outperform presented algorithm in terms of CVN and (ii) are not computationally bound. Therefore to be able to draw definitive conclusions, settings with similar solution quality would have to be compared.

6 CONCLUSION

Within this paper we introduce an improved parallel agent based algorithm for the widely studied VRPTW problem, built around similar concepts as the algorithm presented in [5]. The algorithm is based on the execution of increasingly complex algorithm configurations over a set of instance task orderings.

The main presented contribution is (i) the introduction of an efficient search diversification strategy based on generating a diversified set of orderings using two specific introduced ordering diversification operators and (ii) the presented overall pruning strategy based on three efficient ordering pruning strategies and a vastly improved negotiation process, offsetting the increase in the overall complexity of the algorithm inherent to the diversified search.

The relevance of the improved algorithm is assessed using two sets of widely used benchmark instances. When compared to the state-of-the-art centralized solvers, the algorithm achieves a relative error of 2.4% while equalling the best known results for 64% of the instances. This result also represents a significant improvement over all previously presented agent based algorithm, with 81 new best found solutions. Moreover, benefitting from its inherently parallel nature the algorithm boasts an excellent anytime characteristics, outperforming even the centralized algorithms in this respect.

Future interesting research opportunity was identified in the assessment of suitability of known ordering crossover operators and the eventual applicability of a genetic based approach to identify a fitting ordering for a particular problem instance.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education, Youth and Sports of Czech Republic within the Research program MSM6840770038: Decision Making and Control for Manufacturing III and also within the grant no. LD12044.

REFERENCES

[1] Olli Bräysy and Michel Gendreau, 'Vehicle routing problem with time windows, part I and II', *Transportation Science*, **39**(1), 104–139, (2005).
[2] Zhenggang Dan, Linning Cai, and Li Zheng, 'Improved multi-agent system for the vehicle routing problem with time windows', *Tsinghua Science Technology*, **14**(3), 407–412, (2009).
[3] Klaus Fischer, Jrg P. Mller, and Markus Pischel, 'Cooperative transportation scheduling: an application domain for dai', *Journal of Applied Artificial Intelligence*, **10**, 1–33, (1995).
[4] J. Homberger and H. Gehring, 'A two-phase hybrid metaheuristic for the vehicle routing problem with time windows', *European Journal of Operational Research*, **162**(1), 220–238, (2005).
[5] Petr Kalina and Jiří Vokřínek, 'Algorithm for vehicle routing problem with time windows based on agent negotiation', in *Proceedings of the 7th Workshop on Agents In Traffic and Transportation, AAMAS (to appear)*, (2012).
[6] Petr Kalina and Jiří Vokřínek, 'Parallel solver for vehicle routing and pickup and delivery problems with time windows based on agent negotiation', in *IEEE International Conference on Systems, Man and Cybernetics (to appear)*, (2012).
[7] Robert Kohout and Kutluhan Erol, 'In-time agent-based vehicle routing with a stochastic improvement heuristic', in *11th Conference on Innovative Applications of Artificial Intelligence*. AAAI/MIT Press, (1999).
[8] Hon Wai Leong and Ming Liu, 'A multi-agent algorithm for vehicle routing problem with time window', in *Proceedings of the 2006 ACM symposium on Applied computing SAC 06*. ACM, (2006).
[9] A Lim and X Zhang, 'A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows', *INFORMS Journal on Computing*, **19**(3), 443–457, (2007).
[10] Quan Lu and Maged M. Dessouky, 'A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows', *European Journal of Operational Research*, **175**, 672–687, (2005).
[11] Yuichi Nagata and Olli Brysy, 'A powerful route minimization heuristic for the vehicle routing problem with time windows', *Operations Research Letters*, **37**(5), 333–338, (2009).
[12] P.W. Poon and J.N. Carter, 'Genetic algorithm crossover operators for ordering applications', *Computers and Operations Research*, **22**(1), 135 – 147, (1995).
[13] Sintef. Top — http://www.sintef.no/projectweb/top/problems/.
[14] Marius M. Solomon, 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations Research*, **35**, 254–265, (1987).

Improving Grid Sustainability by Intelligent EV Recharge Process

Mikhail Simonov and **Antonio Attanasio**[1]

Abstract. The paper describes one modeling tool used to simulate the use and recharge of the fleet of Electric Vehicles on local topology served by a number of energy plants supplying the electricity flow. Authors model different categories of users including residents and tourists and real-life events occurring in time dimension. The toolkit attempts the time shift to optimize the use of the available energy flows. It makes the simulation of the growing demand until the sustainability limits of the system will be reached. Therefore the tool might be useful for operational planning and the local government because showing the grid's sustainability limits.

1 INTRODUCTION

The growing demand of electric energy in EU27 is noticeable from the corresponding annual increments in the energy production, as shown in Fig. 1.

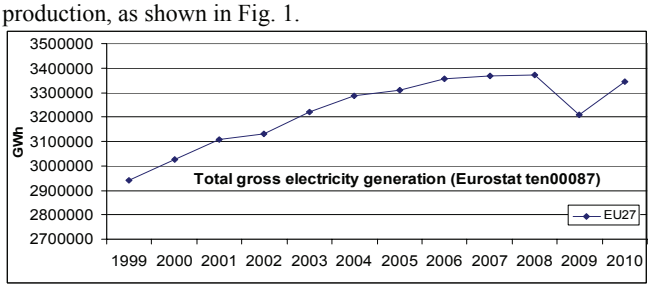

Figure 1. Energy production trend in EU (source: Eurostat).

Except for the -4.9% decrease of electricity production in 2009 caused by the economical crisis, the volume of produced electricity has continuously increased during the last ten years [1]. Even with such a production trend, we expect the sustainability limits will likely be exceeded in the next years, when a high number of Electric Vehicles (EVs) will become ubiquitously available. This is due to both the EVs market, which will undergo a more than linear increase of sales, as shown in Fig. 2, and mostly to the relevant power needed for the recharge of EVs batteries. Residential meters limit energy flows by 3 - 6 kW thresholds. A family with one EV car demands additionally up to 6,25 kW to refill 50 kWh battery in 8 hours. The *quick* recharge schemes are much more energy intensive: 30kW flow (136A at 220V) is required to refill 5kWh in 10 minutes, which corresponds to 10% of the full capacity of 50 kWh EV battery. In fact, while one typical residential house demands up to 14 kWh of electricity daily, an EV energy request stays in a range of 20 – 50 kWh. The simultaneous request for electricity by both residential houses and EVs leads to the significant increase of the global energy demand exceeding the known energy production dynamics. Because of this growing trend, at a certain time the available energy might become insufficient to host the recharge processes of too many EVs.

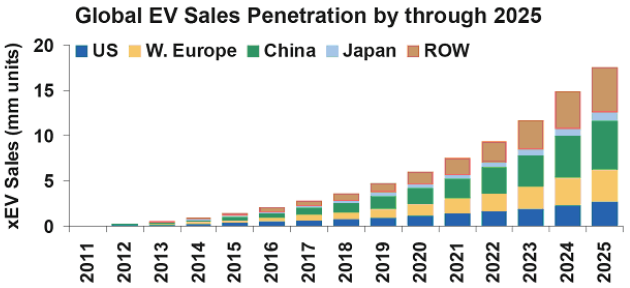

Figure 2. Global EV sales penetration by through 2025 (source: Morgan Stanley Research, North America)

Balancing between the available electricity and its demand is possible from within certain stability limits, which depend on the physical characteristics of distribution sub-topologies, becoming problematic in saturated grids. The work being described is an attempt to propose a toolkit helping to calculate the *sustainability limits* of an electric grid while modeling the real-life processes accompanying the EV use and its recharge.

2 PROBLEM AREA

The reference scenario includes one energy generation capacity characterized by the production load shape function $P_{available}(t) \geq 0$ for any t. The use of renewable energy sources introduces certain time variability impacting negatively in stability terms. An example is less-predictable photovoltaic distributed generation profile. An unexpected production drop in a saturated grid (Fig.3) imposes load shedding actions. It might result in switching off some EV consumers.

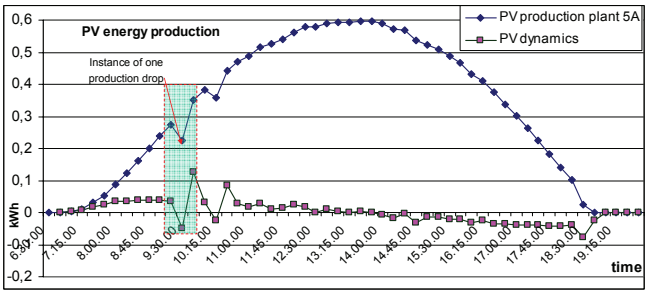

Figure 3. Photovoltaic Energy production profile.

[1] ISMB, Italy, email: {simonov, attanasio}@ismb.it

EV can be considered as a new kind of energy consumer, with an electricity demand profile described by the $EV_i(t)$ function, conventionally assumed as being negative for any t. Depending on its usage by residents and/or tourists, each EV has a mobility profile, which is poorly predictable in time dimension. At the end of mobility stage, the EV_i battery has a residual charge $x_i(t_0)$ denoted also as x_i because of the completeness of the predecessor (mobility process). For a fleet of n EVs, the aggregate energy demand – immediate or delayed one - is expressed by $P_{demand} = (c_1 - x_1) + (c_2 - x_2) + \ldots + (c_n - x_n)$, where c_i is the full battery capacity for EV_i. As the precise mobility profile for each EV is actually unknown, the values of x_i i = [1,n] appear better predictable through statistical distributions. The EV battery recharge process aims to increase x_i values in some way up to c_i. To supply (c_i-x_i) individual energy quantities the grid should provide $(c_i-x_i)/\Delta t$ elementary flow components along time slots t_k, $t_{k+\Delta t}$ and so on.

In energy domain, many researchers ([2], [3], [4]) have produced several load shaping models through the adoption of econometric, statistical, engineering and combined approaches. Load shaping for different categories of users is a highly complex task [4], because it is linked to the imprecisely defined lifestyle and related psychological factors, both subjective by their nature. It is known also that the definition of the standard behavior of the various types of customer in load terms through statistical correlations unlikely can lead to solving the load balancing problem, because it fails to consider the variability of the demand, which is a random factor.

The absolute sustainability limit of the power system, in terms of the overall energy, is expressed by the formula $P_{demand} \leq P_{available}$, while with the time varying expression for a generic instant t, $P_{demand}(t) \leq P_{available}(t)$, we state the importance of maintaining a constant margin of available energy, in order to prevent black-out and to reduce denial of service for clients.

To satisfy the above conditions, different EV battery usages are possible (Fig. 4). Use Case 1 is about saturated grid condition with $P_{demand}(t) > P_{available}(t)$: EV cars wait in the queue until some energy become available. The immediate recharge (Use Case 2) originates Denial-of-Service (DoS) for newcomers when $P_{demand}(t) > 0.99 * P_{available}(t)$. Compared with immediate recharge scheme, the delayed recharge option (Use Case 3) gives a chance to optimize the available energy use by postponing the service for a while. Let denote by *time shifting* the process of moving the immediate energy demand exceeding the grid capacity $P_{demand}(t_k) > P_{available}(t_k)$ to the future time slots t_l with $P_{demand}(t_l) < P_{available}(t_l)$. It reduces the number of DoS occurrences when EV availability for operations is longer compared with the recharge time. More sophisticated energy trading options (Use Cases 4 and 5) give new load management techniques leveraging between the *peak* and *non-peak* periods.

The authors defined four main events belonging to the battery recharge process: *EV_plugging_in* (coming from outside the grid, detected by RFId sensor) , *relè_on_outgoing* (the effective start of EV recharge), *relè_off_outgoing* (the effective end of EV recharge), and *EV_unplug* (from outside the grid). Two additional events (*relè_on_incoming*, *relè_off_incoming*) come from the inverted flow usage known as EV discharge. Until EVs remain plugged into the electricity grid, the intended use of their batteries - from the user viewpoint - is the recharge mode steadily increasing the energy resource up to 100% and then detaching the EVs (Use Cases 2 and 3). From the grid operator viewpoint more modalities are desirable to better manage the demand. For example it could be a delayed or intermittent recharge or even the promiscuous mode combining both recharge and discharge cycles (all Use Cases). In this way, the EVs are transformed in "prosumers" having the $EV_j(t) > 0$ behavior for certain time intervals and $EV_j(t) \leq 0$ for the remaining ones (Use Cases 4 and 5).

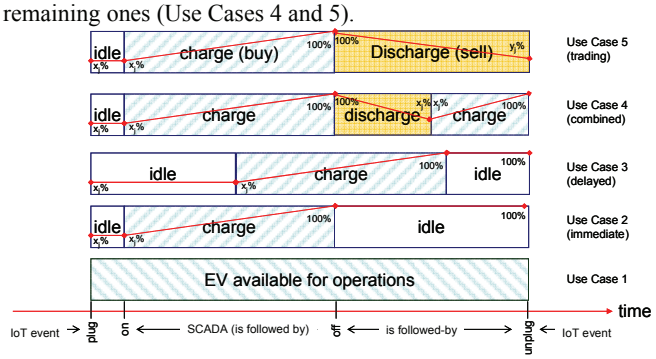

Figure 4. EV battery usage schemes.

The alternation of recharge and discharge processes becomes possible thanks to the presence of the time exceeding that necessary to reach full charge (idle time). Intelligent sequences of recharge-discharge cycles might be used for time-shift purposes, leveraging the peaks on the load shape.

The optimal time-shift combination for all recharge processes could be more easily determined if the usage profiles of all EVs were known. This represents a key feature of the *i-Travel* project (www.ertico.com/i-travel), which has proposed the Travel Agent (Fig. 5) useful to support the use of EV on a local topology, of which the simplest case is an island. The statistics tell that an average time spent in a vehicle is 70 minutes or less. The remaining time EVs are parked. Being attached to the grid, their EV batteries remain available to leverage the energy flows. Supplying the individual travel plans to Travel Agent (TA), it can consider all remaining time as possible candidate timeslots for recharge or time-shift processes.

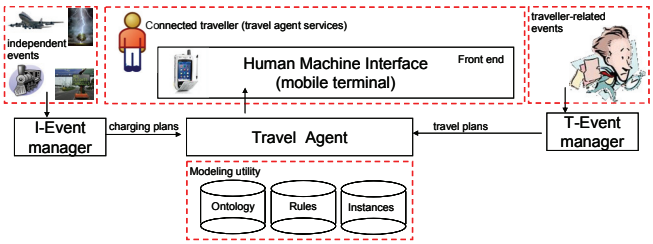

Figure 5. i-Travel project architecture.

Within the considered scenario, the island is populated by a number of residents owning an EV and a number of tourists coming from elsewhere by air and sea travels and getting EVs from local car rental agencies. The external events impacting on the EV usage by tourists are the arrivals and departures of the ships and airplanes (Fig. 6). The publicly available timetables give the estimation of the possible travel plans by tourists renting cars upon their arrivals and releasing them before the definitive departures. On the other hand, the working hours of the residents determine their house-workplace mobility and the consequential statistical distribution of the period of connection to the grid.

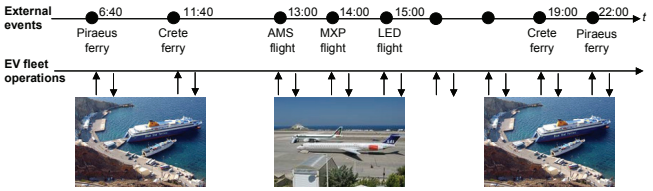

Figure 6. Events occurring on the territory.

The usage of EVs on the island determines the time slots in which they are available for the recharge operations. During the optimization phase, the objective function will result in time-shifting of these time slots to optimize the fleet recharge processes. In order to better formulate the criteria used for the scheduling of recharge processes, a semantic model of the above plans has been developed by defining an ontology and a set of semantic rules, which describe the local topology (an island), the main entities and the relationships of EV use, EV recharge, and electricity supply.

3 MODELING PROCESS

The EV recharge processes can be directly observed by interfacing the intelligent recharge stations (their RFID readers) with the experimental software. An alternative is to model it using simulation tools. The authors designed a model describing EV mobility on an island, prepared software components and used them to evaluate the energy flows in time-space dimensions. Based on the knowledge expressed by domain experts and the timetable of events occurring in sea- and air- ports, the authors have modeled the real life mobility using probabilistic distributions based on the heuristics derived from the observations of the most recurrent events (Fig. 7). In the context of an island, tourists rent EV cars in the aforementioned ports immediately after their arrivals. Tourists leave vehicles some earlier their departure time. The residents use EV cars daily to make home-work-home trips. The set of rules and a number of probability distribution functions were defined and used in the dataset generation/simulation tool. In this way the instances generated accordingly the aforementioned model are conceptually identical to those being supplied by the RFID readers installed on real parking places.

Activity recognition is consistently applied to many human-centric problems [2], to extract the heuristics that simplify the description of the most recurrent daily processes. The understanding and correct statistical modeling of the daily activities is a key element to improve the EV related services. The cause-effect relationship linking the daily activities and the use of the vehicle is then relevant for the car to grid integration. For example, in the proposed model the *end_of_rental* events occur in the port during 60 minutes slot before the ship departure. Similarly, arrivals and departures occurring in airport become *digital events* (DE_j) distributed in 90 min. corresponding slot (Fig. 7). By assuming that after the departing tourists have left their EVs, these are immediately plugged into the grid to recharge their batteries, it is fair to consider a certain correlation between the tourists departures and the time of arrival at the recharge station for their EVs. Residents parking their vehicles in the sea- or air- port areas can plug their EVs for recharge purposes. An additional contribution to the modeling of the EVs use can be obtained from employing both smart metering of load shapes for indoor/domestic activities and GPS/IoT technology to trace the outdoor mobility.

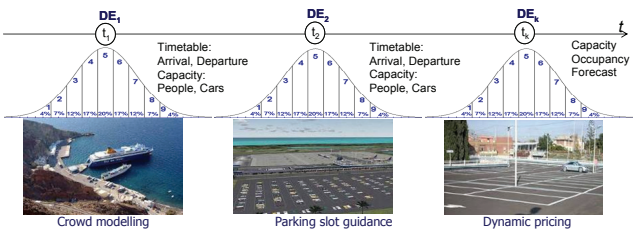

Figure 7. Event space on the local topology.

The major challenge of this work is the optimal balancing of the power flows inside storage-less electric grid by the application of time-shifting (Fig. 8) in the context of a plurality of EVs being charged simultaneously. The First-In First-Out service option is kept until $P_{demand}(t_l) \leq 0.9 * P_{available}(t_l)$. To prevent DoS occurences, the authors attempt time shifting when $P_{demand}(t_l) > 0.9 * P_{available}(t_l)$. When $P_{demand}(t_l) > P_{available}(t_l)$ conditions persist after the optimization attempt, the battery will be recharged at least partially. In the last case the optimization software asks the customer about the expected battery unplugging time (or simulate it), estimates the possible energy quantity as a percentage of the full battery's capacity, and communicates the aforementioned value to the user.

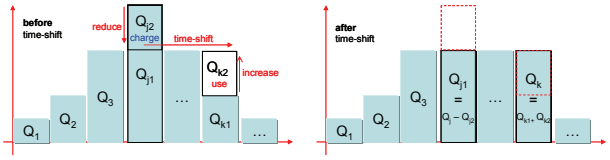

Figure 8. Leveraging load shape using time-shift offered by EV battery

Vehicles equipped by electric engines being interconnected to the electric grid introduce an interesting storage capability and an important advantage in terms of energy that could be temporarily borrowed (or bought) from the EVs connected to the grid. Since a car is used from 30 minutes/day (in UK) up to 97 minutes/day (in Australia) on average, the EV battery could offer to the electricity grid an ubiquitous storage in the remaining time, e.g. up to 22 hours per day (91%). However, in the real life the aforementioned time slot is poorly predictable, calling for the use of modeling.

Since EVs could be recharged ubiquitously and randomly in any geographical location, the symmetry of the system makes unnecessary modeling the directional processes: the number of trips from A to B is typically balanced by a number of trips from B to A. With a capillary presence of a large number of EVs in the electric grid it is possible to exploit this sort of distributed storage in order to delay the recharge of some EVs. On the other hand, it is necessary to manage the queue of recharge processes to avoid that the simultaneous recharge of too many EVs, with a negative impact in grid stability terms.

The study [3], besides showing the importance of simulation, sustains the thesis about the need of a software for identifying and managing the potential impacts of distribution-connected storage. For this reason, starting from the EVs usage model described in this section, the authors have developed a modeling software toolkit, that simulates the EV mobility inside a generic isolated topology together with their recharge processes.

4 MODELING TOOLKIT

The toolkit developed for experimental purposes (Fig. 9) is a collection of algorithms written in C++. The Dataset Generation Tool is a component which generates the EVs arrival and departure events distributed over the topology in both space and time dimension. This module uses both the model and probability distribution functions. Its output is a dataset containing the daily population of individuals using EV cars. In a commercial version, deployable in a real recharge station, the simulated events would be replaced by those generated by RFID sensors, directly interacting with the EVs at the parking places and/or by car rental application software declaring the travel terms/plans mentioned in the rental agreement. Currently, the datasets can be generated on a daily horizon each. The real RFID-based application will stream individuals separately. The batch calculation for different days of the year is also possible. Another software module of the toolkit receives in input the number of days, the expected trend, and it calls several times the Dataset Generation Tool obtaining the collection of daily datasets then. This way the collection of differentiated daily situations accounting the variability of the EV use by human being enables the analysis of trends and sustainability limits of the system.

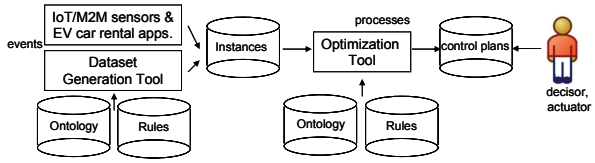

Figure 9. Toolkit architecture.

The variability could be achieved setting up the input parameters such as:

- the number of residents;
- the number of tourists;
- the available energy production (or its load shape function).

The output of the modeling toolkit is the dataset describing the behavior of the dynamic system in time dimension.

The batch component of the toolkit has additional features to simulate time-varying attributes of the evolving population. Namely, it changes dynamically the input parameters to simulate the population dynamics - growth or decline - on the island. As the effect, the ratio between the available energy and the demanded one will be variable as well. It gives a chance to find and/or test the sustainability limits during the whole period.

The data coming from the Dataset Generation Tool are processed from within a module offering the SCADA-like interface to support better the intended use by the grid operator. Besides the other features, the module monitors the balance between available and consumed power flows. It verifies the following constraint:

$$P_{Available} \geq EV_1 + \ldots + EV_n \qquad (1)$$

At certain time the demand might exceed the maximal available power. In such a case the above condition will be false, no more satisfied. The application emanates a warning signal about the dangerous condition in the electricity grid to alert the grid operator then. Finally the optimization tool calculates one “best” control plan thanks to the use of semantic reasoning. It supports and/or automates the decisions that should be taken by grid operator.

5 EXPERIMENTAL RESULTS

The authors have used the Dataset Generation Tool to obtain the daily populations of EV mobility processes describing the EV availability for recharge/discharge. The study started from a simple scenario, such as a little island with 240 residents and 320 tourists only, in which the EV fleet is being used up to the 60% of $P_{Available}(t)$. The authors assumed the $P_{Available}$ setting it to 1000 kW assuming it as a normal operational condition.

The authors evolve the simulation process assuming the increasing demographic trend. The process starts at 100% of the mixed population of residents and tourists and it goes to increase it gradually. It arrives and stops at 150% of the initial values. The obtained collection of data sets is analyzed then. The results are shown on Fig. 10, where the maximum available power remains set to 1000 kW in all cases.

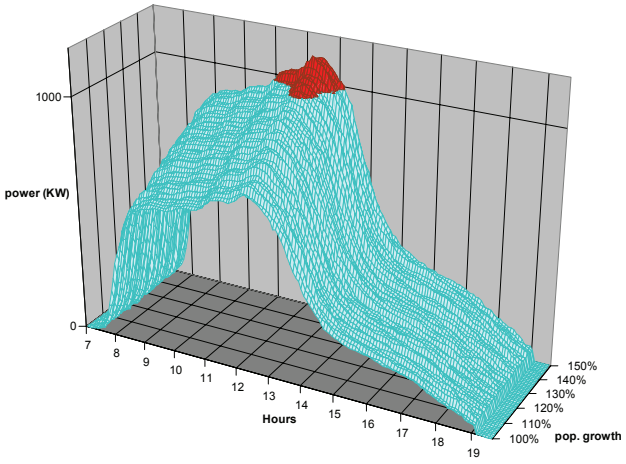

Figure 10. Event space populated by instances of EV recharge processes.

It is worth to notice that, with an excessive population growth, the sustainability condition is broken for certain daytime slots of time only. The red colored peaks on the model show clearly the time periods at which the system looses the sustainability, e.g. the violation of the rule (1).

An example of the *individual* being generated by the toolkit and its corresponding ontological representation are the following ones:

```
Type: TouristEV; Location: Airport_1; Start:
8:51:41; Estimated end: 14:48:1; Charge
power(KW): 2.5; Residual CH(KWh): 5.15229;
Max CHLevel(KWh): 20; Max Power (KW): 20;

<rdf:Description rdf:nodeID="A266">
<evMaxChargePower>20.0</evMaxChargePower>
<evMaxEnergyLevel>20.0</evMaxEnergyLevel>
<evResidualEnergyOnArrival>5.15229</evResidua
lEnergyOnArrival>
<evRequiredPower>2.5</evRequiredPower>
<evEstimatedDepartureTime>14.800278</evEstima
tedDepartureTime>
<evArrivalTime>8.861389</evArrivalTime>
<evPlugsIntoParking #Airport_1"/>
<rdf:type#TouristEV"/>
</rdf:Description>
```

Based on the aforementioned simulation, the authors obtain the indicators as a function of the number of residents and tourists useful to anticipate the sustainability limits. An extract is reported in the Table 1. The energy requests by EVs could be satisfied until the energy demand remains below 1000 kW threshold. For this particular scenario, when both the residential population and tourists will increase more than 30%, the additional energy quantity becomes necessary. It should be introduced by import or considering a building of a new local power plant.

Table 1. The population growth and grid sustainability.

Residents/ Tourists	100%	110%	120%	130%	140%	150%
100%	750,00	772,50	793,75	810,00	827,50	847,50
110%	821,00	840,00	865,00	872,50	895,00	922,50
120%	878,75	870,00	890,00	922,75	938,33	975,00
130%	883,58	905,00	960,00	977,33	**1015,83**	**1034,17**
140%	962,50	995,00	**1014,00**	**1043,75**	**1032,50**	**1070,00**
150%	**1032,50**	**1047,50**	**1085,00**	**1099,17**	**1106,83**	**1129,08**

Let use one dataset containing a traveling population demanding more energy than it is available (Fig.11). The forecast of energy demand is made based on the previous day's data (orange dotted line). Without optimization (blue dotted line), the first DoS occur at 12.00. Applying time shifting (green line), the recharge of some EVs happens outside peak hours.

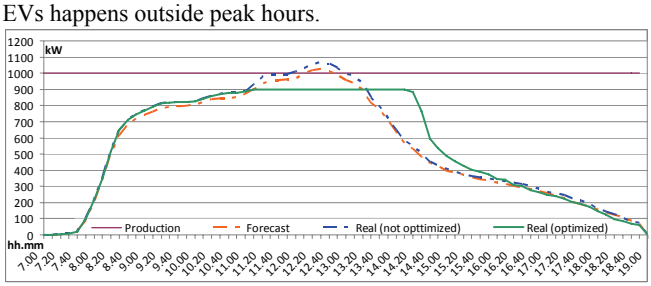

Figure 11. Optimization results: aggregated electricity consumption.

One input dataset is shown on Fig. 12. The optimization results are shown on Fig. 13. The improvement achieved, e.g.; the difference between the original FIFO recharge process and the optimized scheme are visible comparing the Figures 12 and 13.

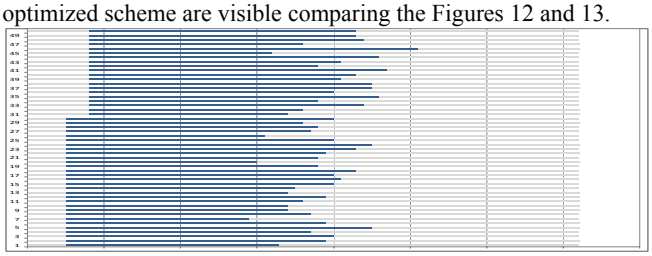

Figure 12. Non optimized FIFO recharge scheme.

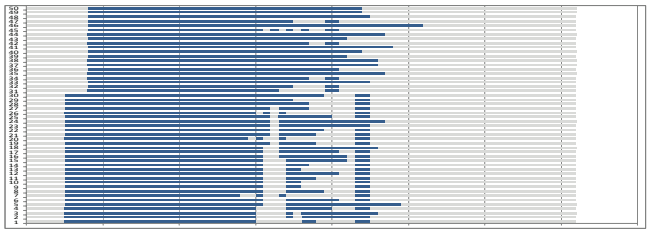

Figure 13. Optimized recharge scheme.

6 ONTOLOGY

As mentioned above, the authors have developed an ontological model of the considered recharge system. This model, together with semantic rules, is used to enhance the management of the scheduling of EVs recharge processes and to constantly monitor the sustainability condition (1).

The ontology describes the main entities and relations between them- EVs and their usage, Power Plants, Power Flows, etc. - that participate to the energy distribution processes through the electricity grid. Afterwards it supports the optimization of charge processes scheduling. In Fig. 14 the principal OWL classes defined within the model are represented graphically.

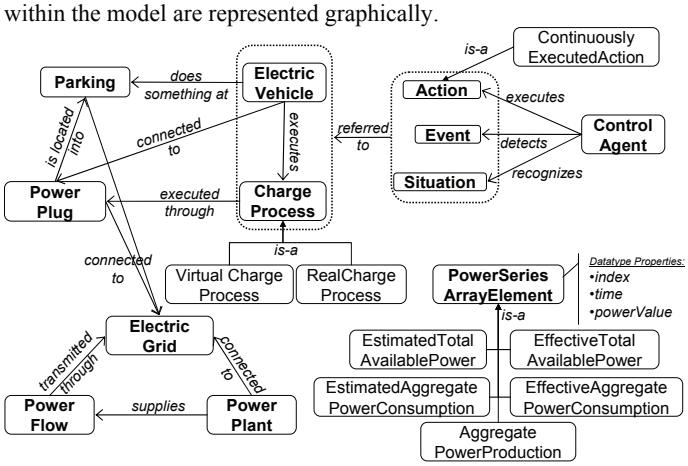

Figure 14. Representation of the ontological model (main classes).

An *Electric Vehicle* connects to a *Power Plug*, which is located inside a specific *Parking*. While charging, it executes a *Charge Process*, which describes, on a time dimension, the evolution of the *Power Flow* used by the EV.

The structure of the *Electric Grid* is deducible from the *connectedOf* properties owned by *Power Plug*, *Parking* and *Power Plant* entities.

Power Series Array Element and its sub-classes are used to model the trend of energy consumption, production and availability over time. More specifically, a discrete time domain is considered, divided in constant intervals (time slots). Each Array Element is part of an hypothetical array which expresses the time evolution of an aggregate power flow. This class owns an *index* property, which uniquely matches the index of the time slot to which it refers. The *powerValue* property is used to express the power intensity (consumed, produced or available) for the corresponding time slot. The ontology model has been encoded in OWL using software tool Protégé. The control module and the rules have been developed using the Apache Jena API.

7 USE OF THE ONTOLOGICAL MODEL

The basic inference mechanism (that simply describes the current state of the system) is executed according to the following assumptions.

While an *Electric Vehicle* is *connectedTo* a *Power Plug*, a *Virtual Charge Process* (sub-class of *Charge Process*) is executed and its power flow is derived from the estimation of the departure time of the EV. Virtual processes are not considered for the sustainability of the system and no reasoning is applied for their power balancing, but they describe the usage of each EV and for

this reason are used to forecast the aggregate energy request (*Estimated Aggregate Power Consumption*) arriving from all EVs.

A *Real Charge Process* implies the presence of a *Power Flow* from the electricity grid. The aggregated energy request coming from these processes at every instant (*Effective Aggregate Power Consumption*) represents the principal problem for the grid sustainability. If we assign negative values to consumed power and positive values to supplied ones (*Aggregate Power Production*), the sum of their values represents, at every time instant, the *Effective Total Available Power*, which should always be greater than zero. Similarly, the *Estimated Total Available Power* is calculated as the sum of *Aggregate Power Production* and (the negative value of) *Estimated Aggregate Power Consumption*. All these classes, representing instantaneous aggregated power flows, are sub-classes of *Power Series Array Element*.

A *Control Agent* is responsible for the processing of an *Event* that occurs into the system. Two major external events (generated by RFId sensors) are considered: *EV_plugging_in* and *EV_unplug*. In practice, a semantic web reasoner is invoked each time an EV is plugged into (or unplugged from) the electricity grid. At each event detection, the *Control Agent*, driven by the reasoner, activates a sequence of rules (recognized *Situations* and executed *Actions*), in order to take the appropriate decisions about the management of EVs charge processes. No other event is considered until the inference cycle ends.

At the end of each time slot, even when no event is detected, another inference cycle is called, in order to update the properties of plugged EVs and to constantly monitor the electrical energy balance (Cyclic Update).

For example, when an EV is plugged into the grid, the reasoner *fires* the appropriate rules in order to decide whether or not to allow the charging. Moreover, a priority ranking is assigned and periodically updated to the EV, based on its residual charge. Since a minimum charge level for every EV should be guaranteed, EVs which have already overcome this threshold receive the lowest priority (e.g. detachable EVs). If an EV has a residual charge lower than the threshold, but, according to its estimated departure time, it is still possible to delay its charge (introduction of idle time before charge process) and reach the minimum charge level, it receives a medium priority ranking (e.g. deferrable EV). When an EV has to be immediately put in recharge in order to reach the minimum charge level, it receives the maximum priority ranking.
When available power runs out, the Control Agent may decide to stop and delay the recharge processes of detachable and deferrable EVs (time-shift), eventually allowing the charging of EVs with the highest priority.

The integration of the renewable energy sources into the smart power grid and piloting the EV recharge processes accordingly the aforementioned methods progress further the known art [7].

8 CONCLUSIONS AND FUTURE WORK

The authors described one new modeling and optimization tool used to simulate the use and recharge of the fleet of Electric Vehicles on local topology served by a number of energy plants supplying the electricity flow. Authors modeled different categories of users including residents traveling between the houses and work places and tourists renting EVs from car rental agencies. The real-life events occurring in time dimension accordingly the model have contributed to determine the EV battery's recharge schemes. Because of the long lasting availability of the plugged EVs - exceeding the minimal needed time to recharge up to 100% level - the toolkit attempts the time shift optimization of the available energy flows at daily basis. It delivers the valid control sequences to govern the interleaved recharge/discharge cycles.

Additionally, the Model Application Utility gives a possibility to run the simulation of the growing demand until the sustainability limits of the system will be reached. Therefore the resulting tool might be useful for operational planning and the local government because showing the grid's sustainability limits.

Based on the experimental results, the authors conclude that the modeling toolkit is useful in the following cases:

- to obtain one dataset describing a day of the EV fleet use to correlate with the expected less-predictable PV production, notably affected by the weather conditions;
- to obtain the sequence of the datasets showing the forecast of the growing trend.

The new optimization method applying time shifting is useful when the sustainability limits are reached. When the known FIFO disciplined recharge processes result in DoS because exceeding the energy limits, the proposed time shifting method moves the recharge of some EVs outside peak hours. The new algorithm kept unused the 10% reserves.

In a future work, the optimization phase will require the use of the *time-shift* feature to better manage the electricity demand by EVs. The authors plan to integrate the Dataset Generation Toolkit by incorporating an efficient algorithm capable to optimize a *massive quantity* of recharge processes.

REFERENCES

[1] Report "Electricity production and supply statistics", Eurostat 2012/5/3, available online at http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Electricity_production_and_supply_statistics, seen on 29/5/2012.

[2] M.L. Chan, E.N. Marsh, J.Y. Yoon, G.B. Ackerman, N. Stoughton, Simulation-based load synthesis methodology for evaluating load-management programs, IEEE Transactions on Power Apparatus and Systems, 4 (PAS-100), 1771--1778, IEEE Press, New York (1981).

[3] J. Broehl, An end-use approach to demand forecasting, IEEE Transactions on Power Apparatus and Systems, 6 (PAS-100), 2714--2718, IEEE Press, New York (1981).

[4] C.W. Gellings, R.W. Taylor, Electric load curve synthesis – A computer simulation of an electric utility load shape, IEEE Transactions on Power Apparatus and Systems, 1 (PAS-100), 60--65, IEEE Press, New York (1981).

[5] E. Kim, S. Helal, and D. Cook, Human activity recognition and pattern discovery, *IEEE Pervasive Computing*, Vol. 9(1), pp. 48-53, IEEE, NJ USA (2010)

[6] P.B. Evans, S. Kuloor, B. Kroposki, Impacts of plug-in vehicles and distributed storage on electric power delivery networks, In Vehicle Power and Propulsion Conference, VPPC '09. IEEE, pp. 838, IEEE, NJ (USA), 2009

[7] M. Liserre, T. Sauter, J. Y. Hung, Future Energy Systems, integrating renewable energy sources into the smart power grid through industrial electronics, IEEE Industrial Electronics magazine, Vol. 4(3) issue 1, pp.18-37, IEEE, NJ (USA), 2010

Application of model-based prediction to support operational decisions in logistics networks

Philip G. Welch[1], **Zsolt Kemény**[2], **Anikó Ekárt**[1] and **Elisabeth Ilie-Zudor**[2]

Abstract. We present a model to predict end of day demand in less-than-truckload freight networks based on continual monitoring of relevant events during the day. Existing advance order information prediction models are extended to exploit the wealth of real world data available in our domain. A simple final model form that can be interpreted by end-users is retained. In a decision support context, human decision makers can assess the predictions and if needed, override the proposed decisions.

Keywords: advance order information, demand prediction, hub-and-spoke networks.

1 INTRODUCTION

The last few decades have seen the emergence of logistics networks that successfully bridge the apparent gap of fast and low-cost shipping of small - typically less-than-truckload[3] (LTL) - consignments. One proven way of such solutions is a multi-level structure that implements, from an operational point of view, a class of hub-and-spoke networks, while from an organisational perspective it is an enterprise network of a "major player" with a large-throughput core structure of *hubs*, contracting smaller local logistics providers also called operating *depots*, with collection and delivery services. The key to the success of such networks is the bundling and re-bundling of shipments that make transporting LTL consignments economically feasible:

- In an *inbound* phase, consignments are collected by a local depot and are shipped to a central hub *bundled by origin*.
- At the hub, consignments are *re-bundled by destination*—the hub is optimised for serving this purpose at a large throughput.
- In an *outbound* phase, vehicles of sub-contracted depots carry consignments bound for their specific destination area.

Typically, the same fleet of vehicles is utilised for inbound and outbound services, i.e., trucks bringing to the hub consignments *from* a given region depart from the hub with consignments *bound for* the same region. This can benefit both types of participants, and enables business models to combine complementary competencies of large-scale efficiency and autonomous functioning with an adequate sharing of obligations and benefits—as also demonstrated in other branches of logistics with related hybrid approaches [11].

Successful as they are, such hub-and-spoke networks operated by multi-level organisational structures still have room for improvement. The major challenges in this context are the *balancing of inbound and outbound loads of a given depot* in order to prevent deadheading vehicles and the coverage of demand bursts with transportation assets. Organisational heterogeneity often inhibits process transparency that would contribute to more efficient operation:

- Subcontractors may use different data models, interpret or handle events in their own specific ways, hampering interoperability needed for seamless network-wide propagation of information.
- Participants, especially subcontractors, may be reluctant to share certain types of operational information as they may compete in local business regardless of collaborating within a larger network.
- De-facto forwarding and reporting practices may result in the information stream lagging behind the material stream, or not leaving sufficient time for well-informed decisions.

Our work is part of the developments ongoing in the EU FP7-funded ADVANCE project, which addresses inefficient utilisation of resources (e.g. unbalanced capacities) in networked logistics companies due to limitations in processing localised information. ADVANCE will support networked companies in improving their information collecting and processing infrastructure, enabling strategic planning coupled with instant decision making. It addresses these problems in two parallel ways: (a) by locating possibilities of improving process transparency and supporting these by solution elements, and (b) by investigating what can be effected within existing transparency limits relying on modelling and prediction, based on known historic data of consignments. A key objective of the project is to allow improvement in a "non-disruptive" way, i.e., new solutions have to fit into existing business models and processes without excessive interference, so that operators can, if necessary, safely revert to conventional practices. This is best done in the form of *decision support* where relevant information is extracted and presented in a human-interpretable way. *It is important to note that the evolvability of such decision support solutions depends on interoperability with human personnel in two ways: (a) the information provided must "make sense", i.e., it must be possible for an operator to interpret it in their daily routine context, and (b) the process leading the decision-critical information must be, to a certain degree, transparent or interpretable as well.*

This paper reports advances in predicting demands based on previously known data, while keeping models simple enough to be interpreted and evaluated by end-users. The paper is organised as follows: first, the targeted scenario is explained, and a formal problem definition is provided, followed by a review of available literature. Hereafter, our specific choices of model and solution approach are put forth in detail, followed by the description of available source data, the experimental setup pursued, and the evaluation of results.

[1] Aston University, Aston Triangle, B4 7ET Birmingham, United Kingdom. E-mail: research@pgwelch.info, a.ekart@aston.ac.uk

[2] Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u. 13–17, H-1111 Budapest, Hungary. E-mail: {kemeny, ilie}@sztaki.hu

[3] LTL consignments vary in size from $\sim$ 100 to $\sim$ 5,000 kg [10].

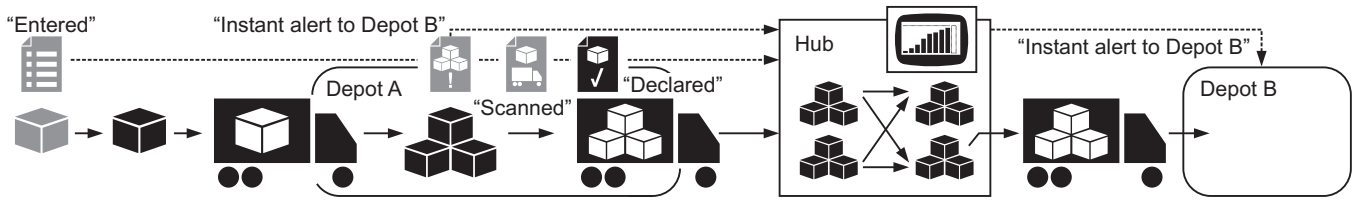

Figure 1. Relevant events of the targeted scenario. Dotted arrows indicate information stream, continuous lines denote material stream.

2 PROBLEM DEFINITION

2.1 Targeted scenario

We focus on a subsequent series of inbound-to-outbound operations from the point of view of a single depot at a time—this boils down to estimating the daily volume of outbound consignments a given depot has to take from the hub by the end of the day. Each consignment has its associated *demand*, i.e., actual volume it occupies within the transportation assets. To this end, we first examine the events of handling a selected consignment (see Figure 1).

Entering a consignment (E). The sender signals the intent of sending a consignment. If this is to be handled via our logistics network, the consignment is *entered* into the system of the depot the sender has contacted. A departure date is, however, not yet specified.

Alerts (A). Should a demand of extraordinary size for a certain destination occur, an *alert* can be sent to the depot in charge of delivery, so that a sufficient number of vehicles is made available to handle the delivery peak. An alert does signal extraordinary demand, but does not necessarily occur for all demand peaks.

Scanning a consignment onto a vehicle (S). Sooner or later, most of the *entered* consignments are prepared for shipment and collected by the depot where they are bundled and dispatched to the hub at an adequate time. Scanning a consignment onto a vehicle (to depart for the hub) is usually recorded shortly before the consignment actually departs, but is not fully guaranteed to be taken to record.

Declaring a consignment. This event signals the departure of the consignment for the hub and is, therefore, the *primary* event for our demand prediction. All events previously mentioned (E, A, and S) can additionally be used for enhancing our forecast but are considered *secondary events*.

Once the inbound vehicles arrive in the hub, consignments are unloaded into the area associated to the depot serving their destination address and this depot takes charge of the consignments. Vehicles of the given depots are then obliged by contract to take all consignments bound for their designated delivery area by a specific deadline. Hence knowing the demand in advance is important: it helps in balancing inbound and outbound traffic, and it is vital to meet outbound obligations once balancing is out of question during demand bursts.

2.2 Addressed problem

With existing transparency limitations, modelling and prediction of demand is of key importance. Here, we investigate making during the same day predictions of the total end-of-day demand for a depot. Our aim is two-fold: to minimise the prediction error and to maximise the simplicity of the final model for better end-user acceptance. A comprehensible model is also advantageous in situations where end-users have access to relevant knowledge that the model may not have (and hence less trust should be placed in the model's prediction)—e.g., knowledge of a local public event that would increase demand.

2.3 Basic notation

Our problem is defined as a regression upon a series $\mathbf{y}$ which accumulates over a day. $y(t)$ is the sum of demand of the consignments declared as being sent by the prediction time t in the current day,

$$y(t) = \sum_j d_j \qquad j \in \{1, \ldots, |D| : \tau_j < t\} \tag{1}$$

where D is the set of all consignments declared by the end of the day and d_j and τ_j are the demand and declaration time for consignment j. If t_e denotes the end of the day we are therefore predicting $y(t_e)$ at time $t < t_e$. We know the distribution of $\mathbf{y}$ for all time in the day earlier than t. We use C as shorthand to denote the current demand $y(t)$ and we denote the remaining demand (that will be declared by the end of the day) as R, where $R = y(t_e) - C$. As C is a known quantity, predicting R is equivalent to predicting $y(t_e)$.

We extend this notation to include both the partially known series from the current day and the multiple completely known series from preceding days. We use $\mathbf{y}_i$ to refer to the series i days previously, hence $\mathbf{y}_0$ is the partially-known series for the current day, $\mathbf{y}_1$ is the completely-known series for the previous day etc. The same convention applies to C and R, hence $C_i = y_i(t)$ and $R_i = y_i(t_e) - y_i(t)$.

3 RELATED WORK

Prediction models which use information on already-booked orders to predict the total number for a period have several names in the literature—predictions using advance order data/information [6, 13], predictions with partial accumulation [4] or advance demand information [12]. We adopt the term *advance order information* (AOI) as it is the more commonly used. The majority of these models make monthly or weekly sales forecasts, to aid planning of inventory and staffing levels. Orders can be placed up to several periods in advance, and hence predictions using AOI are made on this timescale.

Utley and May [13] present an overview of several model types[4]:

- **Additive model**, which makes a prediction $\hat{R}$ of R.
- **Multiplicative model**, which makes a prediction $\hat{y}(t_e) = C/f(t)$, where $f(t)$ is the historical average of $C/y(t_e)$.
- **Combined model** of Kerke et al. [8] where $\hat{y}(t_e) = a_0 + a_1 C$ and a_0 and a_1 are determined from the historic data.

The combined model is equivalent to the multiplicative model when $a_0 = 0$ and the additive when $a_1 = 1$. As $\hat{y}(t_e) = C + \hat{R}$ the combined model can be re-written as an additive model containing C:

$$\hat{R} = a_o + (a_1 - 1)C = a_0 + a_2 C. \tag{2}$$

where we assume a_1 to be constant and absorb it into a constant a_2.

[4] The models listed in [13] make predictions for multiple periods in-advance; as we make same period predictions only we have simplified the notation.

Utley and May also present their own model based on a linear relation between the ratio $y_0(t_e)/y_1(t_e)$ and the ratio C_0/C_1. They do not however offer guidance on extending this model to the seasonal case (where $y_0(t_e)/y_1(t_e)$ will vary systematically across the season). We expect during-week "seasonality" to be present in our data (i.e. different weekdays having different average demand) and hence we do not consider this model to be applicable to our specific case.

Several authors use additive models. Haberleitner et al. [6] detail the implementation of a system in which the prediction $\hat{R}$ is determined by treating R as a time series and applying an exponential smoothing method. Tan [12] presents several simple additive models based upon a combination of human-expert and automatic forecasts. Although suitable for his own application, not all of Tan's models are easily applicable to our own due to (a) their inclusion of expert forecasts, (b) their assumption that the number of orders is small enough to construct a conditional probability of remaining orders based on known ones and (c) their dependence on previous forecasts. The automatically generated component of Tan's Right Tail Estimation method is applicable; similar to Haberleitner this calculates $\hat{R}$ using a time-series approach, in particular a weighted average of previous values of R. Tan also introduces some useful terminology, namely "perfect" and "imperfect" AOI to indicate (a) placed orders that are certain and (b) placed orders that may change before the period end.

Bayesian techniques are also commonly used to address AOI prediction problems [4, 15]. They derive the probability distribution of $y(t_e)$ using probabilistic modelling techniques together with assumed forms for the prior probability distributions of several quantities of interest. Although we can make no comment on the predictive accuracy of these methods, their derived expressions for $\hat{y}(t_e)$ (or the probability of $y(t_e)$) are highly complex and do not meet our requirement of a simple form that can be easily understood by users. We therefore exclude Bayesian approaches from this work.

Comparing our problem to those within the literature we see that although it fits within the domain of AOI predictions, our predictions are on a much shorter timescale which gives us much more training data. If a model is too complex (i.e. has too many fitted terms) for the available training data, over-fitting can occur which degrades performance on out-of-sample data (see Witten and Frank [14]). Conversely if too little complexity is allowed, relevant concepts may be unlearnable. More data allows us to fit models with more terms than those typically found in the literature, potentially increasing prediction accuracy. The trade-off is this makes the model harder for the end-user to understand; hence a balance must be struck.

4 OUR MODEL

We use the combined model in the additive form of Equation 2 as a starting point as this includes both additive and multiplicative models. We augment it by adding the set of attributes shown in Table 1 generated from the partial series $\mathbf{y}_0$ and the recent full series $\mathbf{y}_i, i = 1 \ldots n$. We select a broad set of attributes from which an attribute selection scheme will determine a smaller relevant subset. Following the time series approach we include as attributes the previous n observations $R_i, i = 1 \ldots n$ and the current day of week DW, allowing the model to fit a different constant level per weekday. In addition to C_0 already present in Equation 2, we include the previous n observations $C_i, i = 1 \ldots n$. We also include differenced terms (e.g. $R_i - R_{i+1}$) as time series predictions often use differencing to model trend [3] and their explicit inclusion may allow adjustment for short-term ($\leq n$) trends using a smaller attribute set. Different models are trained for different times of the day, hence the time t is constant within a single trained model and is not needed as an attribute.

By defining a large and diverse initial attribute set we aim to maximise the chances of the final model having a small attribute subset (as it is conceivable that one attribute could convey the same information that would require several in a narrowly-defined set). We make the assumption that the smaller the final model representation is, the easier it will be for non-technical users to understand.

4.1 Extension to multiple source events

The primary event - declaring that a consignment will be sent - is perfect AOI; the demand d_j for that consignment will form part of $y(t_e)$. Within our source data we also have imperfect AOI provided by the secondary events for some, but not all consignments (entering, scanning and alert events as explained before). These occur before the primary event and indicate that the consignment will be sent, but do not specify a date for the primary event. Using Equation 1 we can also derive *secondary series* for each secondary event type and the attributes in Table 1 can be calculated separately for each series. Secondary information is potentially useful for predicting $y(t_e)$, hence it is desirable to include it in our prediction model. In our experiments we test the following inclusion mechanisms:

Secondary attributes. We include the attributes of Table 1 for each secondary series together with the same attributes for the primary series (and therefore regress on multiple cumulative series to predict the final value of one series).

Embedded predictor. We include an embedded predictor—a predicted value for the R of each secondary series—as an input attribute for the primary series prediction. We therefore train a separate model for each secondary series in addition to the primary model.

Waiting consignments. A consignment is in a *waiting* state with regard to a secondary event ϵ if ϵ has occurred (and therefore the consignment is known to the network) but the primary event has yet to occur. From simple inspection of our source data we see the likelihood of a consignment being sent depends on the number of weekdays it has been waiting k. The majority of consignments are sent on the same day or day after, i.e. $k \leq 1$, the number sent dips at $k = 2$, peaks again at $k = 3$, drops at $k = 4$ and those sent for $k > 4$ are negligible.

As it is unclear whether this relationship can be captured in a single attribute for all waiting consignments we define a set of attributes. For each secondary event type ϵ we define the total demand of waiting consignments $w_{\epsilon k}(t)$ based on all secondary events of type ϵ that have occurred for consignments in the previous k weekdays as

$$w_{\epsilon k}(t) = \sum_j d_j \qquad 0 < T_{\epsilon j} \leq k \text{ or } (T_{\epsilon j} = 0 \text{ and } \tau_{\epsilon j} < t) \quad (3)$$

where $T_{\epsilon j}$ and $\tau_{\epsilon j}$ indicate the day and the time of event ϵ for demand d_j for which the primary event has not occurred yet. For all our experiments $k \in \{0, 1, 2, 3, 4\}$ as the consignments with $k > 4$ are negligible.

4.2 Choice of machine learning scheme

Our model is designed to be usable with any machine learning (ML) algorithm that supports regression. We require the trained model to have an easily interpretable representation for the non-technical user.

Assuming a model for each time of day, a real-life implementation may require a different trained model for every minute of the day, hence training cannot require significant computational resources.

Initial experiments performed on a single year of data with different ML algorithms, including linear regression (LR), model trees, neural networks, support vector machines and instance based learners, showed little difference in performance between simpler and more complex methods, with LR sometimes outperforming the latter. As we require the trained model to have an easily interpretable representation and efficient training, LR is the natural choice due to its simplicity and quickly computable closed form expression that can be updated in an online manner as new data becomes available [1].

4.3 Attribute selection and cross-validation

We use an attribute selection scheme to simplify our final model and to guard against overfitting during the training process, which can degrade the performance of ML algorithms (see Witten and Frank [14]). Feature selection methods can be divided into three groups - embedded, filter and wrapper methods (see Guyon and Elisseeff [5]). Unlike the other methods, a wrapper uses the ML algorithm itself as a 'black box' to evaluate the predictive potential of attribute subsets. It is assumed that this evaluation gives a better estimate of accuracy than a separate measure not using the ML algorithm (see Blum and Langley [2]). We therefore choose a wrapper method for our selection scheme and use a simple greedy forward selection mechanism as this is known to be computationally efficient [5].

To ensure that the training data is always chronologically earlier than the independent test data, we use the rolling cross-validation (CV) scheme of Hu et al. 1999 [7] and thus perform a CV without violating causality. The scheme is illustrated in Figure 2. The rolling cross-validation is used within the wrapper method itself. To obtain an independent error measure we nest the entire attribute selection search within an additional outer rolling cross-validation loop (see Kohavi and John [9]). With 10-fold outer cross-validation a single set of model parameters is therefore run 10 times, potentially generating different sets of attributes on each fold.

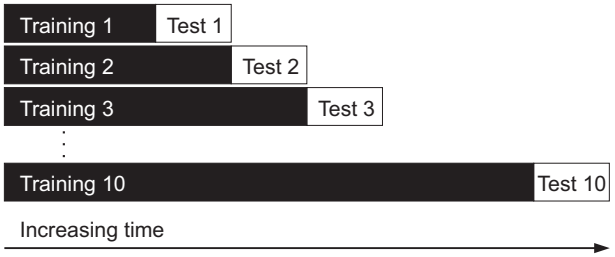

Figure 2. Rolling cross-validation scheme, following Hu et al. 1999 [7]. After each fold the independent test data (in white) is added to the training data. A buffer region (corresponding to the length of Training 1) is included at the start to ensure a minimum amount of data is available for training.

5 SOURCE DATA

We have four years of consignment data from a LTL freight network. Each consignment record has a demand value, the identifier of the delivery depot, the primary event and the E secondary event. The A event occurs for 33% of consignments and the S event for 94%. 86% of consignments have the primary event occurring on the same day as E. For the A event this rises to 90% and for S to 94%. E and A typically occur one or more hours prior to the primary whereas S events typically occur only a couple of minutes before it.

As the current model does not adjust for trend and seasonal effects we limit ourselves to a subset of data where these are minimal. Within the available data we identified a subset of three depots with approximately stationary behaviour over the four years for this purpose. Each of these three depots has $\approx 10^5$ consignment records with ≈ 100 consignments being received on each working day. The mean demand per consignment is ≈ 0.95.

We preprocess the data to remove atypical cases of demand - special cases with zero or near-zero demand (e.g., weekends and public holidays), and also the period from 20 December to 5 January.

Table 1. Candidate attribute set for the single series model only. n was set to 5 for all experiments. As weekends were removed C_5 and R_5 refer to the current day of week in the previous week.

Attribute(s)	Interpretation
DW	Day of week when prediction is made.
$C_i, i \in \{0, \ldots, n\}$	Demand declared by time t and for same time on previous n weekdays.
$R_i, i \in \{1, \ldots, n\}$	Demand remaining to be declared for time t on previous n weekdays.
$(C_0 - C_i), i \in \{1, \ldots, n\}$	Difference between current demand and demand declared on previous weekday, two weekdays before, etc.
$(C_i - C_{i+1}), i \in \{1, \ldots, n-1\}$	Difference between demand declared by time t on consecutive weekdays.
$(R_i - R_{i+1}), i \in \{1, \ldots, n-1\}$	Difference in demand remaining at time t on consecutive weekdays.

6 EXPERIMENTAL SETUP

The attribute selection is run for all three depots and times of day together, generating a single attribute set for each CV fold and model parameters combination. We make predictions at hourly intervals $T = \{12{:}00, 13{:}00, \ldots, 20{:}00\}$ with our limits chosen based on the observation that for all depots $y(t = 12{:}00) \approx 0$ and $y(t = 20{:}00) \approx y(t_e)$. Internally to the selection algorithm, a model is trained for each individual depot and time of day. For each attribute subset examined in the selection we therefore train $3 \times |T| = 27$ models in parallel[5]. The selected attributes are therefore constrained to be identical for each depot and time but the attribute weights can be different.

Before running our main set of experiments we ran the attribute selection scheme to generate an embedded predictor model for each of the secondary series. The data was partitioned to ensure causality; for the embedded predictors the rolling scheme was only run on the first two years of data and in the main experiments only data from the third year onward was used for out-of-sample tests.

6.1 Model parameters

To investigate the impact on predictive performance of including the different event types we ran models including (a) no secondary events, (b) only E (which is earliest and common to all consignments) and (c) all 3 secondary events. The secondary events models were run twice, once with and once without embedded predictors.

[5] $|T| = 9$ as we make predictions at 9 distinct times during the day.

Table 2. Errors for five model configurations. Chosen model is shown in bold typeface. The number of selected attributes is the mean across the 10 folds of the outer cross-validation. The mean of the value being predicted, $R(t_i)$, over all times in day T is 77.13.

Secondary Events	Emb. pred.	No. attribs.	No. sel.	DAMA error	P(Above better)
E,A,S	yes	116	8.5	7.901	N/A
E,A,S	**no**	**113**	**11.8**	**7.907**	**0.552**
E	no	55	10.3	7.938	0.815
E	yes	56	6.0	7.982	0.832
none	no	26	7.3	10.555	1.000

7 RESULTS

All models were run on an 8-core PC with 12 GB of RAM and took around 5 hours computer time in total. We tested the stability of the attribute selection scheme by comparing the different sets of attributes chosen across the 10 outer CV folds. We found that although the sets showed variation after the first couple of attributes selected, the reduction in prediction error for each additional attribute selected was generally small by this point and we therefore expect the impact of any instability upon prediction error to be minor.

At time t we have remaining demand $R(t)$ and our prediction $\hat{R}(t)$. The absolute error at t is therefore $|R(t) - \hat{R}(t)|$. We define our day-averaged mean absolute error (DAMA error) as:

$$DAMA = \frac{1}{|T|} \sum_{t_i \in T} \left| R(t_i) - \hat{R}(t_i) \right| \quad (4)$$

Table 2 lists the DAMA error for each model averaged over all days and depots, ranked by lowest error first. A paired Student t-test was performed between each model and the model ranked directly above it, based on comparing the DAMA error for each depot and day combination in one model against the other. From this we derive the estimated probability that the model in the row above is more accurate than the current row's model (presented in the last column). The model without any secondary series ranks lowest out of all models, with a probability of over 99.9% that its performance is worse than the next-best model. From this we conclude that the inclusion of the imperfect AOI in our secondary series increases prediction accuracy.

Although our top-ranked model has embedded predictors, the probability that it is more accurate than the next-best model is only 55.2%. Also, from models ranked 3rd and 4th, the model with embedded predictor is ranked worse. Thus, we cannot conclude that embedded predictors increase our predictive accuracy. Given that they add significant complexity to the model, we exclude them from our final model selection. If we compare the 2nd to the 3rd ranked model, we see that the inclusion of the S and A series has an 81.5% probability of having improved the prediction error. At the same time, in the different outer CV folds for the 2nd ranked model, the A series attributes were only selected in six out of the ten folds, whereas E and S were always selected. This suggests that the inclusion of S can be expected to be beneficial whereas it is unclear whether the inclusion of series A has any real benefit. Based on this reasoning, we choose the 2nd ranked model including secondary events E, A, S and no embedded predictor as our final model.

7.1 Interpretation of the final model

We performed a run for the final model to select the final attributes using all available data (shown in Table 3). Attributes were selected from the primary, E and S series but not from A. To simplify the interpretation of the final model we group its selected attributes into categories based on the type of source information used to generate them (see Table 3). The categories are:

- **Remaining.** This category is based on the depot's recent history, indicating the remaining demand at the current time over the previous five weekdays for the primary, E and S series.
- **Waiting.** This category is based on current state and indicates the total demand of the currently waiting (not yet declared) consignments which have the E or S event and have been waiting for up to k days.
- **Current.** This is a single attribute C_{E0} based on the current state of the depot signifying the total demand of consignments with an E event occurring by the current time t on the current day.
- **Constants.** The values in this group are based on the entire history of the training data (i.e. the average Monday, Tuesday etc. over all years), constant DW being different for each weekday and constant c common for all weekdays.

We examined the contribution of each category towards the prediction. Although each depot and time-of-day combination is a separate trained model with a separate set of weights for its attributes, a common pattern across all depots which varies systematically over the day is still visible. Figure 3 demonstrates this for three historic depot and day combinations, with one plot per depot and the days spread evenly across both week and year. Categories shown on the $+y$ axis are added to $\hat{R}(t)$ and those on the $-y$ axis are subtracted from it.

At t =12:00 the predictions are based primarily on the remaining and constants categories with waiting making a lesser contribution. As the day continues information about waiting consignments becomes more important until it dominates the prediction. Earlier in the day the model therefore relies primarily on the history before the current day - both the recent and the entire history - but as imperfect AOI becomes available (i.e. more consignments enter the waiting state) the model switches to using these instead.

A proportion of the current (i.e. C_{E0}) is subtracted from the predicted value, with a magnitude that roughly increases over the day. We interpret this as if a larger number of consignments have been entered before time t, a smaller number is expected after t.

Table 3. Selected attributes with explanations for final model, categorised by source information. The attribute set was generated by running the attribute selection on all available data, after the model selection process.

	Remaining (uses short-term history)
R_1,R_4,R_5	Demand remaining at current time on the last weekday, 4 weekdays ago and on same day previous week.
$(R_1 - R_2)$	Difference in remaining at current time on previous weekday and on weekday before that.
R_{E5}	Remaining for *E* series at current time on same day previous week.
R_{S4}	Remaining for *S* series at current time, 4 weekdays ago.
	Waiting (uses current depot state)
(a) W_{E0} (b) W_{E1} (c) W_{E4}	Waiting demand for *E* series consignments waiting for up to (a) current day only (b) current and previous weekday, (c) the 4 previous weekdays inclusive.
W_{S4}	Waiting demand for S series consignments waiting up to the previous 4 weekdays inclusive.
	Current (uses current depot state)
C_{E0}	Current total of E series.
	Constants (uses all history)
DW	Day of the week. A different constant is fitted for each weekday and time of day.
c	A single constant applied to all predictions at the current time.

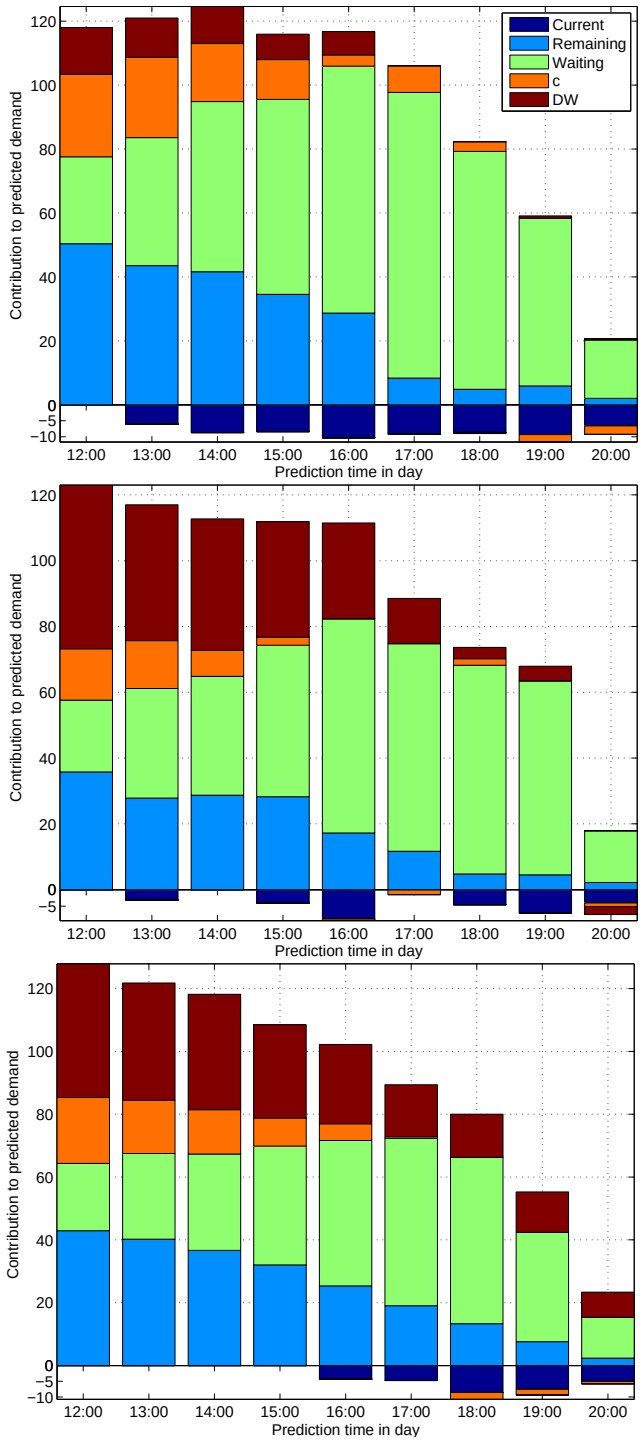

Figure 3. Contribution of the different information categories to the prediction across the day, shown for the final year of data. For the first depot we show the 1st Monday in March (top), for the second depot we show the 1st Wednesday in July (middle) and for the final depot we show the 1st Friday in November (bottom). The y-axis is in units of demand; contributions on the negative y-axis subtract from the predicted number. The members of the constants group (c, DW) are shown separately to highlight their individual contributions.

8 CONCLUSIONS

Although this work focuses in-depth on a case study of a single large collaborative logistics network, several useful points can be made regarding the wider use of AOI models. We have demonstrated a specific scenario where using imperfect AOI clearly increases predictive accuracy, as evidenced by model performance with and without imperfect AOI and the large contribution the imperfect AOI (i.e. waiting) makes in our final model. We have shown that when sufficient training data is available it is relatively straightforward to extend the simple models from the literature to incorporate more information whilst retaining an easily interpretable form. We also note the type of model behaviour observed for our "current" category, where a measure of the current (albeit imperfect) orders was subtracted from the estimate of R, is supported by the combined model of Kekre et al. If circumstances dictated the use of the simple models we would therefore recommend the combined model over a purely additive or multiplicative one.

As maintaining a separate trained model for each time t in the day could be cumbersome in a practical setting, future work should seek to cover the day with a smaller number of models whilst retaining predictive accuracy. Research should also be performed to extend the model to the non-stationary case.

ACKNOWLEDGEMENT

Work presented in the paper was supported by the EU FP7 under Grant No. 257398, "Advanced predictive-analysis-based decision-support engine for logistics" http://advance-logistics.eu/.

REFERENCES

[1] C. Bishop, *Pattern recognition and machine learning*, chapter Introduction, 41, Springer, 1st edn., 2006.
[2] A. Blum and P. Langley, 'Selection of relevant features and examples in machine learning', *Artif. Intell.*, **97**, 245–271, (1997).
[3] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*, Springer, 2nd edn., 2002.
[4] E. De Alba and M. Mendoza, 'Forecasting an accumulated series based on partial accumulation', *J. of Business and Economic Statistics*, **19**(1), 95–102, (2001).
[5] I. Guyon and A. Elisseeff, 'An introduction to variable and feature selection', *J. Machine Learning Res.*, **3**, 1157–1182, (2003).
[6] H. Haberleitner, H. Meyr, and A. Taudes, 'Implementation of a demand planning system using advance order information', *Int. J. of Production Economics*, **128**(2), 518–526, (2010).
[7] Y. Hu, G Zhang, C Jiang, and E Patuwo, 'A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting', *Decision Sciences*, **30**(1), 197–216, (1999).
[8] S. Kekre, T. Morton, and T. Smunt, 'Forecasting using partially known demands', *Int. J. of Forecasting*, **6**(1), 115 – 125, (1990).
[9] R. Kohavi and G. John, 'Wrappers for feature subset selection', *Artif. Intell.*, **97**, 273–324, (1997).
[10] W. Powell, 'A local improvement heuristic for the design of Less-Than-Truckload motor carrier networks', *Transport. Sci.*, 246–257, (1986).
[11] B. Scholz-Reiter, D. Lappe, C. Ruthenbeck, and C. Toonen, 'Introduction of a hybrid control approach for automotive logistics', in *Proc. 11th WSEAS int. conf. on robotics, control and man. tech.*, ROCOM'11/MUSP'11, pp. 69–73, (2011).
[12] T. Tan, 'Using imperfect advance demand information in forecasting', *Ima J. of Management Mathematics*, **19**, 163–173, (2008).
[13] J. Utley and J. May, 'The use of advance order data in demand forecasting', *Operations Management Res.*, **3**, 33–42, (2010).
[14] I. Witten and E. Frank, *Data Mining - Practical Machine Learning Tools and Techniques*, Elsevier, 2nd edn., 2005.
[15] J. Xiangrong, L. Xiongjian, and C. Yaxi, 'A new method for short time series forecasting', *China Communications*, (2009).

Safety Stock Placement in Non-cooperative Supply Chains

Péter Egri[1]

Abstract. The paper studies the safety stock placement problem in decentralised supply chains consisting of autonomous stages. For the inventory optimisation problem we apply the guaranteed-service model, while the non-cooperative attitude is handled with mechanism design theory. We propose and investigate four different mechanisms based on the Vickrey – Clarke – Groves scheme, and their distributed implementation. We illustrate on numerical examples how the mechanisms achieve the globally optimal solution in different ways.

1 INTRODUCTION

In order to provide high service levels for the customers, companies have to maintain inventories, and these are accumulated at the most expensive point of the supply chain as end-products [6]. For example, in the U.S. automotive sector recently so much finished cars have been kept in inventories, that they would have been enough for satisfying average demand for 60 days [1]. Japanese auto manufacturers perform significantly better, e.g., Toyota keeps finished goods to cover demand for a 30 days shorter period than General Motors.

European automotive companies face similar problems. Customers expect to have their orders fulfilled in a couple of days, and they demand very high service levels [6]. Recently, several efforts have been made in order to cope with this challenge by innovatively applying modularity, flexibility, lead-time reduction [12] and collaborative planning [3]. These solutions deal with technologies, short-, medium-, and long-term planning, lean production, but the global supply chain design optimization is often missing.

Inventory positioning is such a strategic issue in complex supply networks—like the one indicated on Fig. 1—that aims at minimising overall inventory cost, while guaranteeing a given service level for the customers. There are examples from the automotive industry for 30% reduction in inventory levels after repositioning of the inventories, while at the same time, preserving the high standards of the service [16]. In [11] it is mentioned that usually 25-50% reduction in holding cost is achievable, and the inventory positioning is illustrated on some large-scale industrial examples.

However, the applicability of such global optimisation approaches in distributed environments requires cooperative attitude, i.e., that the participants agree on minimising the total costs. This may be—although not necessarily—true in the supply network of a single company, but almost inconceivable in a network consisting of different companies.

Global optimisation problems involving agents with different goals can be successfully handled by *mechanism design* theory, which facilitates the alignment of conflicting goals with the global objective. In this paper we combine an inventory positioning model with mechanism design analysis in order to extend the applicability of strategic supply chain design methods across companies.

Figure 1. A part of an automotive supply network.

The remainder of the paper is organised as follows. In Section 2 we overview the related literature. Next we present the optimisation model for serial chains, and investigate four different mechanisms that can achieve the optimal solution in Section 3. We demonstrate the differences of the mechanisms using a numerical study in Section 4. Finally, in Section 5, we conclude the paper and enumerate some possible future research directions.

2 LITERATURE REVIEW

Mechanism design theory deals with the problem of constructing the rules of a game with incomplete information in order to achieve some preferred outcome. It assumes an independent, benevolent decision maker, who collects the private information from the agents, decides about the outcome, and pays to the agents for disclosing the private knowledge.

One of the main achievements in this field is the Vickrey – Clarke – Groves (VCG) mechanism, which is the only one in the general model that can provide *efficient* (globally optimal) and *truthful* (agents are not interested in lying about their private information) behaviour. Nisan and Ronen combined the classic mechanism design theory with computer science considerations in their seminal paper, where they also illustrated the application of the VCG mechanism on the shortest path problem [13]. It was later proved that despite the advantageous truthfulness and efficiency properties of the presented mechanism, it tends to overpay the agents, and the overpayment can be arbitrary large [5]. Recently, algorithmic mechanism design has been extensively used in multiagent optimization problems, such as multiagent planning [20] and resource allocation [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, email: egri@sztaki.hu

Implementing a mechanism without an independent decision maker is the field of *distributed mechanism design* [14, 15]. In [7] a general method called *replication* is presented for implementing VCG mechanisms in a distributed way. However, it does not solve the problem of *budget-balance*: an independent source for the agents' payments is still required.

Applying mechanism design for supply chain optimization is not completely new in the literature. Both [8] and [9] present different decision problems in decentralised supply chains, and they apply the VCG mechanism for solving them. However, both models assume an independent decision maker, and do not investigate the possibilities of distributed implementation. In [4] a two-stage supply network is considered with a single supplier and multiple retailers, and a combined mechanism design and information elicitation model is developed. In this special case, a non-VCG mechanism can be applied, which is truthful, efficient, budget-balanced, and can be implemented in a distributed way, resulting in a theoretical model for the Vendor Managed Inventory (VMI) business practice.

A recent review of general inventory control models in supply chain management can be found in [18]. The problem of safety stock placement in supply chains is discussed in [11], where two different approaches, the stochastic- and the guaranteed-service models are presented. In this paper, we adopt the latter one, and repeat its solution method in the simplest case, considering a serial supply chain in Section 3.1.

3 MODEL

In this section we investigate a strategic supply chain design problem, the safety stock placement, in a non-cooperative setting with rational agents. We consider a serial supply chain with n stages, where the nodes represent manufacturing or transportation operations as shown in Fig. 2. Inventory can be held after each node with different h_i unit holding costs. The market demand is stochastic, but the T_i processing lead-times at the nodes are deterministic. We assume that there is no fixed ordering or setup cost, and the nodes apply a *base-stock policy*: an order for stage i immediately generates an order with the same quantity towards stage $i+1$ in order to maintain the base-stock level. We also assume the *guaranteed-service model*: guaranteed service time S_i means that if stage $i-1$ places an order in period t, it receives the goods in period $t+S_i$, and the service time for the final customers is given as a boundary condition ($S_1 = s_1$).

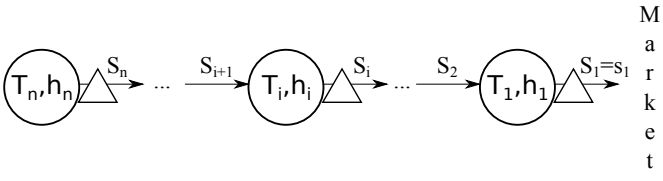

Figure 2. Supply chain setting.

3.1 Centralised approach

The demand in each period is assumed to be independent, normally distributed random variable with mean μ and standard deviation σ. Thus the total demand of t consecutive periods is normally distributed with mean μt and standard deviation $\sigma\sqrt{t}$. The required inventory for satisfying the demand of t periods is therefore $\mu t + k\sigma\sqrt{t}$, where μt is the expected demand and $k\sigma\sqrt{t}$ is the safety stock. The k safety factor should be determined depending on the allowed probability of stock-out, $1-\alpha$, where α denotes the required *service level*. Table 1 shows the appropriate safety factors for some α values (based on [17]). It is assumed that the demand over t periods cannot exceed $\mu t + k\sigma\sqrt{t}$ (or else it is lost, backlogged, served from an other source or with extraordinary production).

Table 1. Service level and safety factors.

α	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%	99.9%
k	1.28	1.34	1.41	1.48	1.56	1.65	1.75	1.88	2.05	2.33	3.09

Guaranteed service time $S_i \geq 0$ means that if stage $i-1$ places an order in period t, it receives it in period $t + S_i$. With the processing lead-time, the *replenishment time* at stage i will be $S_{i+1} + T_i$, where it is assumed that $S_{n+1} = 0$. If stage i wants to provide service time S_i, it therefore needs to hold inventory for $S_{i+1} + T_i - S_i$ periods, which is called the *net replenishment time*.

Since negative net replenishment time is meaningless, we have the constraints $S_i \leq S_{i+1} + T_i$. From this limitation also follows, that if S_1 should equal to s_1, the minimum service time at stage i is s_1 minus the total lead-times in the chain $(i-1, \ldots, 1)$. Let us define the minimum service time as

$$\underline{S}_i = \max\left\{0, s_1 - \sum_{j=1}^{i-1} T_j\right\}, \tag{1}$$

thus we have the constraint $\underline{S}_i \leq S_i \leq S_{i+1} + T_i$.

Using the net replenishment time, the base-stock level at stage i can be calculated as

$$B_i = \mu(S_{i+1} + T_i - S_i) + k\sigma\sqrt{S_{i+1} + T_i - S_i}, \tag{2}$$

and the expected inventory in period t becomes

$$\mathbb{E}[I_i(t)] = B_i - \sum_{j=0}^{t-S_i} \mu + \sum_{j=0}^{t-S_{i+1}-T_i} \mu = k\sigma\sqrt{S_{i+1} + T_i - S_i}. \tag{3}$$

The total expected inventory holding cost for the supply chain is

$$\sum_{i=1}^{n} h_i k\sigma\sqrt{S_{i+1} + T_i - S_i}, \tag{4}$$

therefore the optimal service times can be determined with the following non-linear program:

$$\min \sum_{i=1}^{n} h_i\sqrt{S_{i+1} + T_i - S_i} \tag{5}$$

s.t.

$$\underline{S}_i = \max\left\{0, s_1 - \sum_{j=1}^{i-1} T_j\right\} \quad i \in \{1, \ldots, n\} \tag{6}$$

$$\underline{S}_i \leq S_i \leq S_{i+1} + T_i \quad i \in \{1, \ldots, n\} \tag{7}$$

$$S_1 = s_1 \tag{8}$$

$$S_{n+1} = 0 \tag{9}$$

Let $P(T, h, s_1)$ denote the above program with the lead-time and cost vectors T and h. Note that $P(T, h, s_1)$ is independent from μ, σ and k. This means, that whether a stage should hold inventory or not is independent from the specific mean and variance of the demand, and can be determined only with the knowledge of the lead-times and holding costs.

Simpson proved that the minimum of the objective function occurs at a vertex of the convex polyhedron defined by the constraints of the program [19]. This means that in an optimal S^*, each S_i^* equals either $S_{i+1}^* + T_i$ (when the stage does not hold any inventory) or $\underline{S}_i$ (where the stage offers immediate—or minimal—service time). Based on this result, Graves and Willems showed that the problem can be solved efficiently using the following dynamic programming recursion [10]:

$$f_{n+1} = 0 \tag{10}$$

$$f_i = \min_{j=i+1\ldots n+1} \left\{ f_j + h_i \sqrt{\underline{S}_j + \sum_{l=i}^{j-1} T_l - \underline{S}_i} \right\} \quad (i \le n) \tag{11}$$

where f_i is the optimal cost in the $(n, n-1, \ldots, i)$ chain if stage i holds safety stock for providing $\underline{S}_i$ service time.

3.2 Mechanism design for safety stock placement

Let us consider the case, when the stages of the supply chain are independent, rational entities with private information, i.e., h_i and T_i are only known at stage i. Instead of minimising the total cost, each stage intends to minimise its own cost, which can be done simply by not holding stock at all, except at stage 1, which has to keep an enormous end-product stock in order to guarantee service time s_1.

The solution for this situation provided by the mechanism design theory is to assume a central decision maker, who collects the private information from the stages, determines the service times and provides some payment t_i for each stage, see Fig. 3. Since the stages might distort their disclosed information, we denote the inventory holding costs and lead-times collected by the mechanism as $\hat{h}_i$ and $\hat{T}_i$. The utility function of the stages becomes

$$u_i = t_i - v_i(S) \tag{12}$$

where t_i is the payment received, and $v_i(S) = h_i k \sigma \sqrt{S_{i+1} + T_i - S_i}$ is the expected inventory holding cost at stage i. (When $S_{i+1} + T_i - S_i < 0$ then $v_i(S) = 0$.)

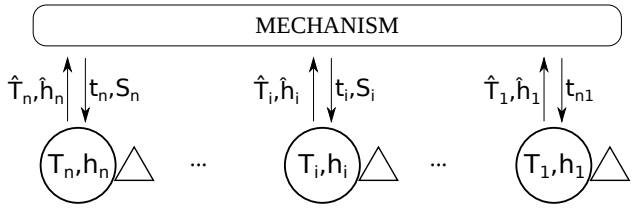

Figure 3. Mechanism design setting and information flow.

A VCG mechanism applied to the safety stock placement problem determines S^* as the solution of $P(\hat{T}, \hat{h}, s_1)$, and defines the payments in the form of

$$t_i = g_i(\hat{h}_{-i}, \hat{T}_{-i}) - \sum_{j \ne i} \hat{v}_j(S^*), \tag{13}$$

where $\hat{h}_{-i} = (\hat{h}_1, \ldots, \hat{h}_{i-1}, \hat{h}_{i+1}, \ldots, \hat{h}_n)$, $\hat{T}_{-i} = (\hat{T}_1, \ldots, \hat{T}_{i-1}, \hat{T}_{i+1}, \ldots, \hat{T}_n)$, g_i is an arbitrary function independent from $\hat{h}_i$ and $\hat{T}_i$, and $\hat{v}_i(S) = \hat{h}_i k \sigma \sqrt{S_{i+1} + \hat{T}_i - S_i}$.

It is well known, that VCG mechanisms are *truthful*, consequently, the stages can optimise their utility by disclosing $\hat{h}_i = h_i$ and $\hat{T}_i = T_i$. Furthermore it is *efficient*, viz., it minimises the total holding cost in the chain. The functions g_i give the freedom for constructing different mechanisms, e.g., $g_i \equiv 0$ results in a situation, where each stage must pay the total cost of the chain minus its own. In the next subsections we examine different VCG mechanisms and their properties.

The general idea that we use for developing specific VCG mechanisms is the following. We change $\hat{h}_i$ and $\hat{T}_i$ values in $\hat{h}$ and $\hat{T}$ for a predetermined $\tilde{h}_i$ and $\tilde{T}_i$, thus the resulting vectors denoted as $\hat{h}^{(i)}$ and $\hat{T}^{(i)}$ will not depend on $\hat{h}_i$ and $\hat{T}_i$. Then we calculate an optimal $S^{(i)}$ solution for the program $P(\hat{T}^{(i)}, \hat{h}^{(i)}, s_1)$, and define the g_i function as

$$g_i(\hat{h}_{-i}, \hat{T}_{-i}) = \sum_{j \ne i} \hat{v}_j(S^{(i)}). \tag{14}$$

Let $\tilde{v}_i(S) = \tilde{h}_i k \sigma \sqrt{S_{i+1} + \tilde{T}_i - S_i}$ denote the expected inventory holding cost function in the modified problem for stage i. The next theorem characterises the payment of any such mechanism.

Theorem 1 $\tilde{v}_i(S^*) - \tilde{v}_i(S^{(i)}) \ge t_i \ge \hat{v}_i(S^*) - \hat{v}_i(S^{(i)})$

Proof Since S^* minimises the objective function of $P(T, \hat{h}, s_1)$

$$\hat{v}_i(S^*) + \sum_{j \ne i} \hat{v}_j(S^*) \le \hat{v}_i(S^{(i)}) + \sum_{j \ne i} \hat{v}_j(S^{(i)}), \tag{15}$$

which can be rearranged resulting $t_i \ge \hat{v}_i(S^*) - \hat{v}_i(S^{(i)})$.

On the other hand, $S^{(i)}$ minimises the objective function of $P(\hat{T}^{(i)}, \hat{h}^{(i)}, s_1)$, therefore

$$\tilde{v}_i(S^{(i)}) + \sum_{j \ne i} \hat{v}_j(S^{(i)}) \le \tilde{v}_i(S^*) + \sum_{j \ne i} \hat{v}_j(S^*), \tag{16}$$

thus we get $\tilde{v}_i(S^*) - \tilde{v}_i(S^{(i)}) \ge t_i$. □

The theorem provides an upper and a lower bound on the payments, which helps to characterise the expected utility of the stages as well as the budget of the mechanism (the total payment). Note that in some of the mechanisms we apply infinite $\tilde{h}_i$ modified holding cost, in which cases the upper bound also becomes infinite, thus fails to provide any useful information about the possible overpayment. However, due to the definition, the payments are always finite.

3.2.1 *Commonly known lead-times ($\mathcal{M}_1$)*

Firstly, we consider the situation where only the holding cost (h_i) is private information at stage i, and the T_i values are common knowledge. Although this contradicts our original assumptions, we decided to include this case for providing a comparison with the further mechanisms.

We use $\tilde{h}_i = \infty$ and the commonly known T in $P(T, \hat{h}^{(i)}, s_1)$. We further assume that $s_1 \ge T_1$, otherwise the stage 1 would have to keep some safety stock in order to guarantee s_1 service time, and then any feasible solution would be optimal—with infinite total cost. Since in the modified program the holding cost at stage i is infinite, in the optimal $S^{(i)}$ solution stage i will not carry any safety stock, i.e.,

$S_i^{(i)} = S_{i+1}^{(i)} + T_i$. This mechanism is analogous to the shortest path mechanism [13]: stage i receives as large payment as its contribution to the cost decrease of other stages.

The next theorem states, that if a stage does not hold inventory, it also does not receive any payment, otherwise it gets compensation which is not less than its cost.

Theorem 2 *If $S_i^* = S_{i+1}^* + T_i$ then $t_i = 0$. Else $S_i^* = \underline{S}_i$ and then $t_i \geq \hat{v}_i(S^*)$.*

Proof Since $S_i^{(i)} = S_{i+1}^{(i)} + T_i$, Theorem 1 in this case means $\tilde{v}_i(S^*) \geq t_i \geq \hat{v}_i(S^*)$, from which the statement follows. □

An immediate corollary of the theorem is that $\forall i : u_i \geq 0$, which property is called *individual rationality*. Furthermore, the mechanism has deficit, i.e., $\sum_{i=1}^n t_i \geq 0$.

Unfortunately, if T_i is private information this mechanism cannot be applied, since $S^{(i)}$ would then depend on $\hat{T}_i$, which corrupts the properties of the VCG mechanism. Therefore, in the following subsections we construct different mechanisms for the case when T_i is private information.

3.2.2 Disregarding holding costs ($\mathcal{M}_2$)

In this mechanism we use $\tilde{h}_i = 0$ and an arbitrary $\tilde{T}_i$. Since in this modified problem the inventory holding is free in stage i, it will keep as much safety stock, as possible ($S_i^{(i)} = \underline{S}_i^{(i)}$). Furthermore, none of the upstream stages holds any stock ($S_j^{(i)} = S_{j+1}^{(i)} + \hat{T}_j$, $(j = i+1, \ldots, n)$), and therefore $S_{i+1}^{(i)} = \sum_{j=i+1}^n \hat{T}_j$. It can be seen that the optimal $S^{(i)}$ is indeed independent from the value of $\tilde{T}_i$. This mechanism corresponds to the *Clarke pivot rule*, and the following theorem characterises its properties. The theorem follows from Theorem 1 using $\tilde{T}_i = 0$.

Theorem 3 $0 \geq t_i \geq \hat{v}_i(S^*) - \hat{h}_i k\sigma\sqrt{\sum_{j=i+1}^n \hat{T}_j - \underline{S}_i^{(i)}}$

This mechanism has surplus, i.e., $\sum_{i=1}^n t_i \leq 0$. This approach can be interpreted as comparing the optimal S^* solution to $S^{(i)}$, where stage i holds maximal inventory. It can be seen that this is unfair to the lower stages, where the possible maximal inventory is larger.

3.2.3 Disregarding lead-times ($\mathcal{M}_3$)

The next mechanism is constructed by using $\tilde{h}_i = \infty$ and $\tilde{T}_i = 0$. In the optimal $S^{(i)}$ stage i will not hold any stock, and therefore $S_i^{(i)} = S_{i+1}^{(i)}$. The payment in this case can be characterised by the following theorem (corollary of Theorem 1).

Theorem 4 *If $S_i^* = S_{i+1}^* + \hat{T}_i$ then $0 \geq t_i \geq -\hat{h}_i k\sigma\sqrt{\hat{T}_i}$. Else if $S_i^* = \underline{S}_i$ then $t_i \geq \hat{v}_i(S^*) - \hat{h}_i k\sigma\sqrt{\hat{T}_i}$. In the special case when $S_i^* = \underline{S}_i = 0$ then $t_i \geq 0$.*

This mechanism can work either with surplus or deficit, thus it can be viewed as a transition between the previous two mechanisms.

3.2.4 Considering average lead-times ($\mathcal{M}_4$)

In this subsection, we try to approximate the behaviour of the mechanism $\mathcal{M}_1$ by defining $\tilde{h}_i = \infty$ and $\tilde{T}_i = \sum_{j\neq i} \hat{T}_j/(n-1)$, i.e., the mean lead-time of the other stages. If we assume that $s_1 \geq \tilde{T}_1$, then the optimal $S^{(i)}$ solution satisfies $S_i^{(i)} = S_{i+1}^{(i)} + \tilde{T}_i$, wherewith Theorem 1 is reduced to the following form.

Theorem 5 *When the lead-time of stage i is below or equal to the average ($\hat{T}_i \leq \tilde{T}_i$), then $t_i \geq \hat{v}_i(S^*)$.*

Otherwise, when the lead-time is above or equal to the average, and $S_i^* = S_{i+1}^* + \hat{T}_i$ then $0 \geq t_i \geq -\hat{h}_i k\sigma\sqrt{\hat{T}_i - \tilde{T}_i}$, else $t_i \geq \hat{v}_i(S^*) - \hat{h}_i k\sigma\sqrt{\hat{T}_i - \tilde{T}_i}$.

The corollary of the theorem is that the stages are interested in decreasing their lead-times, since decreasing it below the average guarantees non-negative utility.

3.2.5 Summary of the mechanisms

In the next table we summarise the construction of the previous four mechanisms.

Table 2. Summary of the mechanisms

	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$
$\tilde{h}_i$	∞	0	∞	∞
$\tilde{T}_i$	T_i	*	0	$\text{avg}_{j\neq i}\, T_j$

3.3 Decentralised protocol

In order to implement a mechanism in a decentralised way, i.e., without a trusted centre, two issues should be addressed: (i) the computation of the optimal safety stock placements, and (ii) assuring the payment for each stage. The first issue can be resolved by *replication*, which is a standard technique for implementing a VCG mechanism in a decentralised setting *faithfully*, i.e., in such a way, that the rational stages are not interested in deviating from the proposed protocol [7]. For example, if the stages disclose their private information to all of the other stages (so every stage knows $\hat{h}$ and $\hat{T}$), and then all of them can compute the optimal service times and payments. If they agree on the solution, they adopt it, otherwise they suffer a severe penalty, e.g., by missing the opportunity of serving the market.

Regarding the second issue, we suggest that the payments of the mechanism have to be covered from the market. Let us consider the situation presented on Fig. 4, where predefined unit prices p_i for the product and components are given.

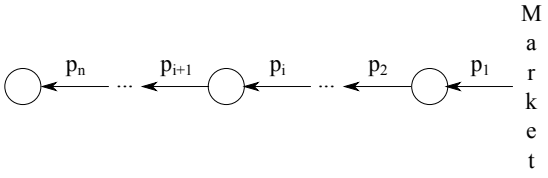

Figure 4. Decentralised implementation of the mechanism.

In order to assure the appropriate payment for the stages, the unit prices have to be modified. Note that we assume that the modification of p_1 does not influence the demand. This can be assumed if the modification is sufficiently small, therefore we prefer mechanisms that imply small change to p_1.

Let us define the new unit prices as $p_i' = p_i + \sum_{j=i}^n t_j/\mu$. With this modification, the expected utility of stage i—disregarding any

production cost—in each period becomes

$$u_i = (p'_i - p'_{i+1})\mu - v_i(S^*) = (p_i - p_{i+1})\mu + t_i - v_i(S^*) \quad (17)$$

therefore the proposed decentralised implementation keeps the truthfulness and efficiency of the VCG mechanisms.

4 COMPUTATIONAL STUDY

4.1 Numerical example

Table 3 (page 6) illustrates the results of using the four different mechanisms in a supply chain with 10 stages. The parameters of the problem are $s_1 = 5$, $\mu = 1000$, $\sigma = 100$ and $k = 2.05$. The h_i, T_i and p_i parameters are indicated in the table. In the optimal case, stages 2 and 6 keep safety stock. In accord with the theorems, the mechanism $\mathcal{M}_1$ assigns payment only to those two stages, and the payment is not less than the expected holding cost. Mechanism $\mathcal{M}_2$ determines only negative payments, except for stage 6, which is the uppermost stage holding stock. The third and fourth mechanism assign both positive and negative payments as well; and in the latter case, the non-negative payment for stages with lead-time below the average (3.6) can also be observed. Note that we have disregarded the production costs in the model, which would only cause a constant shift in the utilities, therefore do not influence neither the optimal solution nor the payments. That is the reason of the unexpected increase of the utility at the uppermost stage.

Fig. 5 illustrates the same costs and the payments according to the different mechanisms at each stage graphically.

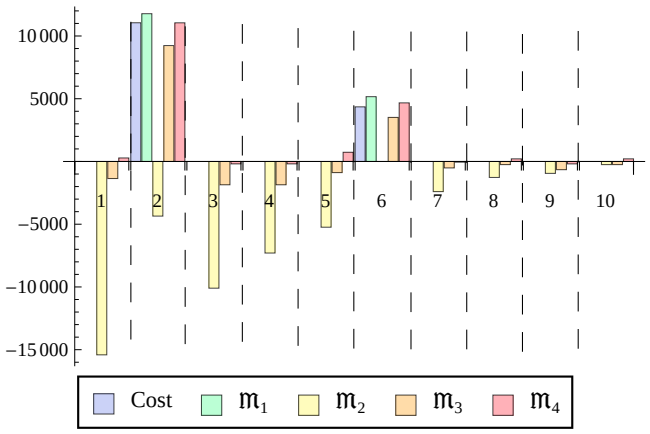

Figure 5. Illustration of the costs and payments at the different stages.

Table 4 shows the total payments which can be compared to the total holding cost of the optimal S^*. It can be seen that $\mathcal{M}_3$ resulted in a total payment closest to zero, and therefore it caused the smallest change in the market price by increasing it with only 2% .

4.2 Simulation

In order to check which mechanism results in the least change of the market price, we have run several experiments with different parameters. Table 5 shows the average results based on 500 simulation runs. The n, s_1, k, σ, μ and p parameters were the same as in the previous example, while the lead-times and holding costs were randomly generated in each run, but using the same dataset for each mechanisms. The h_i values are from a uniform distribution with support $[n - i + 1, 3(n - i + 1)]$, which is reasoned with the observation that holding cost is likely to be higher downstream the supply chain. The T_i parameters were generated from an uniform distribution over $\{1, \ldots, 5\}$. The generating approach of the lead-times simulate various combinations of long manufacturing and short assembly operations, as well as long transportation times from global (e.g., Far Eastern) suppliers.

Table 4. Total payment and change in the market price

	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$
$\sum v_i(S^*)$	15410			
$\sum t_i$	16944	-47276	5125	16526
p'_1/p_1	1.075	0.79	1.02	1.075

Table 5. Average performance of the mechanisms based on 500 runs.

	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$
$\sum v_i(S^*)$	13535			
Avg $\sum t_i$	17773	-48628	2416	18133
Avg p'_1/p_1	1.036	0.901	1.005	1.037

It can be seen that mechanism $\mathcal{M}_1$ results in relatively high total payment. A corollary of Theorem 2 is that the total payment can not be less than the total cost. However, no upper bound was given for the payment, and thus the problem of overpayment may occur, similarly to the case of the shortest path mechanism [5]. The mechanism $\mathcal{M}_4$ approximates the first mechanism by using an average lead-time instead of the real one in the payment calculations, and therefore they resulted in similar behaviour. The $\mathcal{M}_2$, which allows only non-positive payments, results in an enormous negative payment, which is more than three times bigger than the inventory holding cost itself. The decentralised protocol in this case results in approximately 10% decrease in the market price, which—assuming price-independent demand—is clearly not desirable for the supply chain. Finally, $\mathcal{M}_3$ resulted in a fairly low total payment, and its indicated increase in the market price was only 0.5%.

5 CONCLUSIONS AND FUTURE WORK

We investigated the safety stock placement problem in non-cooperative serial supply chains, motivated mainly by the inventory management problems of global automotive supply networks. We applied mechanism design theory combined with the appropriate operation research models for minimising the overall inventory holding cost. We presented and compared four specific mechanisms based on the VCG scheme, and examined their distributed implementation.

There are several possible extensions of this work. Besides the presented mechanisms several others are possible, including randomised ones that may have more desirable properties. Providing upper bounds on the total payment, proving *approximate budget-balance*, is also an important research direction. Considering price-dependent demand leads to a more complex, but more realistic inventory holding and pricing problem. *Group-strategyproofness*, i.e., preventing collusions in possible coalitions, would also worth further investigations. A distributed implementation with partial information sharing, where the agents do not share complete private information with every other agents would make the model much more practical.

Table 3. Numerical example.

	i	1	2	3	4	5	6	7	8	9	10
	h_i	16	15	24	10	10	5	8	4	2	1
	T_i	3	5	4	4	2	5	4	2	5	2
	S_i^*	5	2	10	6	2	0	13	9	7	2
	$v_i(S^*)$	0	11056	0	0	0	4354	0	0	0	0
	p_i	223	159	114	81	58	42	30	21	15	11
$\mathcal{M}_1$	t_i	0	11778	0	0	0	5166	0	0	0	0
	$t_i - v_i(S^*)$	0	722	0	0	0	812	0	0	0	0
	p_i'	240	176	119	87	63	47	30	21	15	11
	u_i	63780	46279	32541	23243	16602	12671	8471	6050	4322	10804
$\mathcal{M}_2$	t_i	-15410	-4354	-10099	-7297	-5240	0	-2401	-1275	-950	-249
	$t_i - v_i(S^*)$	-15410	-15410	-10099	-7297	-5240	-4354	-2401	-1275	-950	-249
	p_i'	176	128	86	64	48	37	25	19	14	11
	u_i	48370	30147	22442	15946	11362	7505	6070	4775	3371	10555
$\mathcal{M}_3$	t_i	-1359	9239	-1857	-1857	-886	3510	-514	-249	-654	-249
	$t_i - v_i(S^*)$	-1359	-1816	-1857	-1857	-886	-844	-514	-249	-654	-249
	p_i'	228	166	111	80	59	43	28	20	14	11
	u_i	62421	43741	30684	21387	15716	11015	7956	5801	3668	10555
$\mathcal{M}_4$	t_i	280	11051	-191	-191	732	4671	-54	210	-192	210
	$t_i - v_i(S^*)$	280	-5	-191	-191	732	317	-54	210	-192	210
	p_i'	240	176	119	87	64	46	30	21	15	11
	u_i	64060	45552	32350	23053	17334	12176	8417	6260	4129	11014

We emphasise that combining planning models with the results of the algorithmic mechanism design can be applied to different logistic problems; the model presented in the paper is only one example. Therefore considering more complex planning problems is also a possible future working field.

ACKNOWLEDGEMENTS

This work has been supported by the OMFB No. 01638/2009 grant and the János Bolyai scholarship No. BO/00659/11/6. The author also thank József Váncza for his help and support.

REFERENCES

[1] G.P. Cachon and M. Olivares, 'Drivers of finished-goods inventory in the U.S. automobile industry', *Management Science*, **56**(1), 202–216, (2010).

[2] M. de Weerdt, Y. Zhang, and T. Klos, 'Multiagent task allocation in social networks', *Journal of Autonomous Agents and Multi-Agent Systems*, **25**(1), 46–86, (2012).

[3] 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*, **4**(3), 226–233, (2011).

[4] P. Egri and J. Váncza, 'Supply network coordination by vendor managed inventory – a mechanism design approach', in *Proc. of the 2nd Workshop on Artificial Intelligence and Logistics (AILog-2011)*, pp. 19–24, (2011).

[5] E. Elkind, A. Sahai, and K. Steiglitz, 'Frugality in path auctions', in *Proc. of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 701–709, (2004).

[6] *From build-to-order to customize-to-order: Advancing the automotive industry by collaboration and modularity*, eds., R. Ericsson, R. Becker, A. Döring, H. Eckstein, Th. Kopp, I. Poslu, and J. Váncza, The AC/DC Consortium, ISBN 978-91-633-6973-5, 2010. Code of Practice Findings of the EU-FP6 Project AC/DC – Automotive Chassis Development for 5-Days Cars.

[7] J. Feigenbaum, M. Schapira, and S. Shenker, 'Distributed algorithmic mechanism design', in *Algorithmic game theory*, eds., N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani, Cambridge University Press, (2007).

[8] M. Feldmann and S. Müller, 'An incentive scheme for true information providing in supply chains', *Omega*, **31**, 63–73, (2003).

[9] D. Garg, Y. Narahari, E. Foster, D.M. Kulkarni, and J.D. Tew, 'A Groves mechanism approach to decentralized design of supply chains', in *Proc. of the IEEE Conference on Electronic Commerce*, pp. 330–337, (2005).

[10] S.C. Graves and S.P. Willems, 'Strategic safety stock placement in supply chains', in *Proc. of the 1996 MSOM Conference*, Hanover, NH.

[11] S.C. Graves and S.P. Willems, 'Supply chain design: Safety stock placement and supply chain configuration', in *Supply chain management: Design, coordination and cooperation*, eds., A.G. de Kok and S.C. Graves, volume 11 of *Handbooks in Operations Research and Management Science*, 229–339, Elsevier, (2003).

[12] W. Menzel, J. Lentes, A. Döring, R. Ericsson, and L. Siljemyr, 'Dynamic supply loops – a concept for flexible and faster automotive supply network management', in *Advanced Manufacturing and Sustainable Logistics*, volume 46 of *Lecture Notes in Business Information Processing*, 130–140, (2010).

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

[14] D.C. Parkes and J. Shneidman, 'Distributed implementations of Vickrey-Clarke-Groves mechanisms', in *Proc. of 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 261–268, (2004).

[15] J. Shneidman and D.C. Parkes, 'Specification faithfulness in networks with rational nodes', in *Proc. of the 23rd Annual ACM Symposium on Principles of Distributed Computing*, pp. 88–97, (2004).

[16] D. Simchi-Levi, *Operation Rules – Delivering Customer Value through Flexible Operations*, MIT Press, 2010.

[17] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, *Designing and managing the supply chain: Concepts, strategies, and cases*, McGraw – Hill, New York, 2000.

[18] D. Simchi-Levi and Y. Zhao, 'Performance evaluation of stochastic multi-echelon inventory systems: A survey', *Advances in Operations Research*, **2012**, 34 pages, (2012). Article ID 126254, doi:10.1155/2012/126254.

[19] K.F. Simpson, 'In-process inventories', *Operations Research*, **6**, 863–873, (1958).

[20] 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)*, pp. 423–427, (2008).

Towards Decentralised Agent-Based Inter-Company Scheduling

Haixiao Liu and **Jeroen Keppens** and **Michael Luck**[1]

Abstract. This paper examines inter-company scheduling in scenarios where the companies wish to collaborate with one another but are not prepared to reveal operations or sales information. For example, this type of scenario occurs when multiple small to medium sized manufacturers of the same or similar products wish to pool their resources to meet larger orders even though they remain competitors with one another. Conventional approaches to inter-company scheduling are inappropriate in such situations because they rely on companies to reveal potentially sensitive information to one another or to a third party. This paper presents a novel fully decentralised agent-based approach to inter-company scheduling that substitutes information sharing for negotiation. The performance of this approach is assessed in a range of simple scenarios consisting of only two companies.

1 INTRODUCTION

It is often advantageous for separate organisations to coordinate the planning and scheduling of their activities for a variety of reasons. For example, inter-company scheduling can allow the organisations involved to operate more cost-effectively, enable them to adapt to environmental changes more quickly or achieve objectives that they could not achieve otherwise [3][16]. Various multi-agent systems for inter-company scheduling have been proposed and these have achieved considerable success [12][7].

Conventional approaches to agent-based inter-company scheduling approaches assume that organisations are prepared to share access to potentially sensitive private information, such as orders and production capacity, with a third party entity. For example, a yellow page agent is introduced in [8] to maintain the overall capacity of involved companies. Although these approaches to inter-company scheduling are effective in many scenarios, there are circumstances where the organisations that may benefit from inter-company scheduling tend to be unwilling to share as much information with a third party as these approaches require [15][4].

We envision a scenario consisting of a number of small and medium sized manufacturers that produce similar products. On the one hand, these organisations compete with one another because they manufacture similar products. As such, much of the information that is shared with a third party in conventional inter-company scheduling approaches is commercially sensitive. Therefore, the manufacturers in our scenario are unwilling to share such information. On the other hand, in certain situations, small and medium sized manufacturers can benefit from pooling resources with their competitors to achieve economies of scale or, simply, to allow them to fulfil larger orders. This scenario requires a different paradigm of inter-company scheduling: one where resource sharing is accomplished through bilateral negotiation between the organisations involved. A potential drawback of such an approach is that globally optimal scheduling solutions cannot be achieved due to lack of information sharing.

We propose a novel fully decentralised agent-based approach to inter-company scheduling that substitutes information sharing for negotiation. This approach effectively tackles the problem for two reasons. Firstly, manufacturing companies can independently make beneficial decisions to allocate their own resources. Secondly, since manufacturers do not depend on another or a third party to share resources, they may choose not to disclose private information for their own good. Therefore, our approach can solve the inter-company scheduling problem in a decentralised manner with privacy protection.

This paper presents a first step into our investigation to decentralised agent-based inter-company scheduling, by means of a simplified scenario that involves only two manufacturers. The scenario is defined in Section 2. Next, Section 3 proposes a novel fully-decentralised agent-based inter-company scheduling approach that can deal with the scenario introduced in 2. This approach is evaluated in Section 4, where we examine to what extent our approach under-performs due to the organisations' unwillingness to share information. Section 5 reviews related approaches to inter-company scheduling are reviewed. Finally, our future work towards decentralised agent-based inter-company scheduling is described in Section 6.

2 SCENARIO

Our scenario is depicted in Figure 1. It consists of two manufacturers 1 and 2, each aiming to maximise its profit. Each manufacturer receives orders for the products it can manufacture. It also obtains income by fulfilling orders. A company's production capacity is restricted by the production resources it possesses. We make the following assumptions about inter-company scheduling:

- Order: Only one type of product is required in all customer orders. Each order is not fulfilled unless the manufacturer produces as many products as stipulated in the order. No income is received for orders that are only partially fulfilled.
- Resource: Only one type of resource is requested for production. A company can increase the number of resources available for production by buying additional resources from the other company. Conversely, a company that sells resources to the other reduces the resources available to it.
- Production: There is a linear relationship between resource input and product output.

[1] Department of Informatics, King's College London, United Kingdom, email: haixiao.liu@kcl.ac.uk

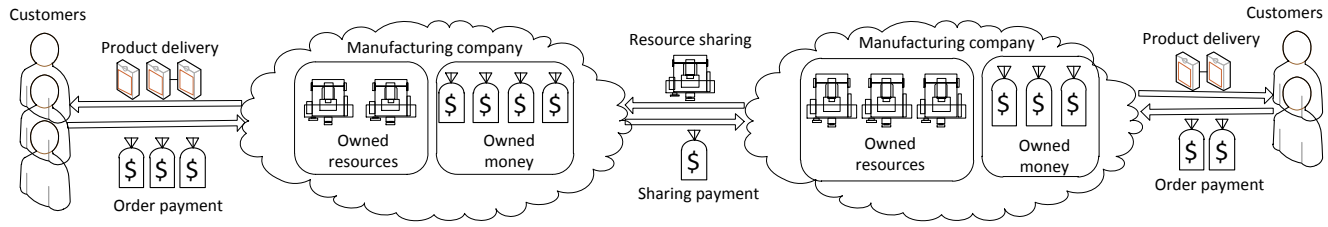

Figure 1. Our scenario

Let d be the market price for the product. The manufacturer i owns c_i units of resources and receives a set of orders O_i from customers. Each order $q \in O_i$ represents the quantity of the product required to fulfil the order. We assume that to manufacture one unit of the product, m_i units of resource are required in manufacturer i. For each order $q \in O_i$ that the manufacturer i fulfils, $q * m_i$ units of resources are allocated and an income $q * d$ is received.

A number of additional variables are introduced to represent scheduling solutions and their implications. The boolean variable ω_q describes whether or not the order $q \in O_i$ is accepted. The positive integer variable a_q expresses the number of resources assigned to order q; and the integer variable s_i indicates the number of units of resource bought or sold in manufacturer i. p_i is the price paid for sharing each unit of resource between two manufacturers. Here positive or negative values of s_i indicate different meanings as below.

- $s_i > 0$: the manufacturer i intends to purchase s_i units of resource.
- $s_i < 0$: the manufacturer i wants to sell $-s_i$ units of resource.
- $s_i = 0$: the manufacturer i does not wish to buy or sell any resources.

In our scenario, when the resource sharing occurs there is a buyer and a seller, which leads to $s_1 = -s_2$. Even when both manufacturers do not intend to share resources, this equation still holds as $s_1 = 0$ and $s_2 = 0$.

An allocation of a_q resource to production to meet order q is represented as a pair (ω_q, a_q). Let A_i be the set of all of i's production resource allocations. Then, a scheduling solution, or schedule, for manufacturer i can be expressed as a vector $x_i = \langle A_i, (s_i, p_i)\rangle$. The profit associated schedule x_i is denoted $P(x_i)$. The optimal assignment X_i is the most profitable solution of allocating resources for processing customer orders based on its own production capacity, ignoring any potential for buying or selling units of resource. Its profit P_i can be calculated from Equation 1.

$$P_i = \max\{\sum_{q\in O} q * d | O \subset O_i, \sum_{q\in O} q * m_i \leq c_i\} \tag{1}$$

If a rational manufacturer is unable to meet all its orders, it is willing to purchase them provided the cost of purchase is less than the marginal profit realised from those resources. In other words, a manufacturer is only willing to purchase resources that increases total profit. Thus, from the point of resource buyer i, then the following two constraints apply to s_i and p_i:

$$\begin{array}{l} 0 \leq s_i \leq \sum_{q\in O_i} q * m_i - c_i \\ 0 \leq p_i \leq ((s_i + c_i) * d/m_i - P_i)/s_i \end{array} \tag{2}$$

A self-interested company will only agree to sell resources instead of committing them to production if the former yields a higher overall profit than the latter. The amount of resources it can sell cannot exceed its capacity. Moreover, the profit after selling should not be less than allocating those sold resources for production or letting them remain unused. Then from the resource seller i's point of view, two constraints in an acceptable schedule x_i should be met as in Equation 3.

$$\begin{array}{l} s_i \geq -c_i \\ P(x_i) \geq P_i \end{array} \tag{3}$$

The negotiation strategy on resource sharing is used in manufacturer i to propose the quantity s_i and price p_i. In resource sharing negotiations, a prospective buyer would like to purchase resources to meet as many orders as possible for maximal income from production. Moreover, for the same amount of resource that it intends to purchase, the buyer is willing to pay as little as the seller agrees for minimal expenses.

We assume for s_i units of resources, the prospective buyer initially desires to pay b_i of marginal profit. Then the initial purchase request for prospective buyer is decided as in Equation 4.

$$\begin{array}{l} s_i = \sum_{q\in O_i} q * m_i - c_i \\ p_i = (\sum_{q\in O_i} q * d - P_i) * b_i/s_i \end{array} \tag{4}$$

The resource purchase request of manufacturer i should be updated when (s_i, p_i) is not agreed by the other. It can gradually increase the payment by g_i of marginal profit as in Equation 5 when the payment p_i does not reach the upper bound, namely $p_i < ((s_i + c_i) * d/m_i - P_i)/s_i$.

$$\begin{array}{rl} s'_i = & s_i \\ p'_i = & \min\{((s_i + c_i) * d/m_i - P_i)/s_i, \\ & p_i + ((s_i + c_i) * d/m_i - P_i) * g_i/s_i\} \end{array} \tag{5}$$

If the payment reaches the marginal profit, namely $p_i = ((s_i + c_i) * d/m_i - P_i)/s_i$, the prospective buyer may reduce the amount of resources it is willing to purchase. This process of updating (s_i, p_i) to (s'_i, p'_i) is specified in Equation 6.

$$\begin{array}{rl} s'_i = & \max\{\sum_{q\in O} q * m_i | O \subset O_i, \\ & \sum_{q\in O} q * m_i < s_i + c_i\} - c_i \\ p'_i = & (s'_i + c_i) * d/m_i - P_i)) * b_i/s'_i \end{array} \tag{6}$$

The objective of companies is to maximise their profit from fulfilling orders and sharing resources as formalised in Equation 7.

$$\max \sum_{q\in O_i} \omega_q * q * d - s_i * p_i \tag{7}$$

In order to achieve this objective, the manufacturer has to satisfy the resource and process constraints. The former one as shown in Equation 8 means that the amount of resources assigned to orders and shared with the other company should not exceed its capacity. Equation 9 indicates the process constraint that any order is not met unless the complete needed resources are allocated. In addition, the valid range of variables are specified in Equation 10. X_i can be obtained by using these equations with $s_i = 0$ and $p_i = 0$.

$$\sum_{q \in O_i} a_q - s_i \leq c_i \tag{8}$$

$$\forall q, a_q \geq \omega_q * q * m_i \tag{9}$$

$$\omega_q \in \{0, 1\}; a_q \in N. \tag{10}$$

3 APPROACH

In our scenario, each manufacturer is an autonomous decision maker. For this reason, each manufacturer is represented by an agent in our approach. Following the scenario, an agent i is defined by a tuple $\langle c_i, m_i, O_i \rangle$, where c_i represents i's resource capacity, m_i i's production process and O_i is the set of customer orders of i.

Each agent must make decisions on how to allocate resources. A unit of resource can be allocated to production, sold to the other agent or remain unused. An agent may also decide to attempt to increase the number of resource units available to it by purchasing them from its competitor and negotiate a price for this transaction. It is assumed that each agent is driven by profit maximisation.

The decision making process of a single agent in the decentralised agent-based inter-company scheduling system is shown in Algorithm 1. The algorithm consists of two phases.

In the first phase, an agent i searches for the optimal resource assignment X_i through the simplex algorithm. The profit P_i of this assignment is used as the lower limit for any decisions regarding resource sharing. If an agent is unable to fulfil all its orders by means of the resources it possesses, then it generates an initial resource purchase request $rq = (i, s_i, p_i)$. This request contains a proposal for the quantity of required resources and payment, both of which are decided according its negotiation strategy as in Equation 4. The simplex algorithm is utilised here for corresponding resource allocation for production A_i as well.

In the second phase, the agents negotiate potential resource sharing. In our simple scenario, the way in which the negotiation proceeds depends on the circumstances of the agents. If neither of the two agents sends a purchase request, both commit to allocating their resources following the optimal assignment X_i. If one sends a purchase request and the other does not, then further rounds of negotiation take place. The recipient of the message answers with a reply $rp = (answer, rq)$ where $answer \in \{accepted, declined\}$. It declines the proposal rq if agreeing to it does not enable it to increase its profit. This decline leads to the sender updating its request according to its negotiation strategy as in Equations 5 and 6. This continues unless the negotiation terminates. If both agents make rent requests, then they simultaneously send out their purchase requests until the negotiation terminates. Here agents only send out requests to purchase resources and replies to agree or disagree to the corresponding requests.

The negotiation process terminates when either an agreement is reached or when the agents discover it is not possible to reach an agreement. The latter situation occurs when an agent reaches one of its boundary conditions during the negotiation process as in Equation 2.

Algorithm 1 Decentralised agent-based scheduling approach

```
sharing_agreed = false
P_i = get_profit_bound() //calculate the profit lower bound as in Equation 1
X_i = get_schedule(0, 0) // use the simplex algorithm to find out the schedule X_i and its profit P_i in Equations 7, 8, 9 and 10 with s_i = 0 and p_i = 0
if Σ_{q∈O_i} q * m_i < c_i then
    s_i = initialise_quantity()
    p_i = initialise_payment() // set variables as in Equation 4
    x_i = get_schedule(s_i, p_i)
    P(x_i) = get_profit(s_i, p_i) // use the simplex algorithm to find out the schedule x_i and its profit P(x_i) in Equations 7, 8, 9 and 10 with just initialised variables s_i and p_i
    rq ← (i, s_i, p_i)
else
    x_i ← X_i, P(x_i) ← P_i, rq ← (i, 0, 0)
end if
repeat
    send(rq)
    receive(msg)
    if msg is the reply rp = (answer, rq) then
        if answer = accepted then
            commit to x_i
            sharing_agreed = true
        else
            s_i'' = update_quantity(s_i, p_i)
            p_i'' = update_payment(s_i, p_i) // update variables as in Equations 5 or 6
            send(rq'' = (i, s_i'', p_i''))
            if s_i'' = 0 then
                commit to x_i
                sharing_agreed = true
            end if
        end if
    else if msg is the request rq' = (i', s_i', p_i') then
        if s_i = 0 ∧ s_i' = 0 then
            commit to x_i
            sharing_agreed = true
        else
            s_i' ← −s_i', p_i' ← −p_i'
            x_i'= get_schedule(s_i', p_i')
            P(x_i') = get_profit(s_i', p_i')
            if P(x_i') ≥ P(x_i) then
                commit to x_i'
                sharing_agreed = true
                send(rp = (accepted, rq'))
            else
                if s_i = 0 then
                    send(rp = (declined, rq'))
                else if s_i > 0 then
                    s_i'' = update_quantity(s_i, p_i)
                    p_i'' = update_payment(s_i, p_i)
                    send(rq'' = (i, s_i'', p_i''))
                end if
            end if
        end if
    end if
until sharing_agreed
```

4 ANALYSIS

In our scenario, a manufacturer may be able to increase its profits through resource sharing. However, the extent to which resource sharing is possible is constrained by the amount of information that the companies are willing to share. In this section, we compare the performance of our approach (denoted DS or decentralised scheduling) with two alternative approaches:

- Local scheduling (LS): the two agents do not exchange any information and do not share any resources as a result. In other words, two agents are isolated entities and no negotiation occur between them. This approach constitutes a baseline on the performance of DS.
- Global scheduling (GS): the resources and orders of both agents are pooled into a single agent and its optimal resource allocation is computed. In our scenario, this approach is equivalent to full information sharing. It constitutes an upper bound on the potential performance of DS.

This paper examines the effect of different features of our scenario on overall profitability under DS, compared to LS and GS, by generating random instances of our scenario on a set of input parameters. The effects of the following five input parameters are examined:

- Overall demand (OD): the total volume of products required by customers of both companies. Formally,

$$\sum_{i=1,2} \sum_{q \in O_i} q = OD \tag{11}$$

- Demand distribution (DD): the ratio of the whole demand of one company to that of the other company. Formally, $DD = x : y$ iff

$$\frac{\sum_{q \in O_1} q}{\sum_{q \in O_2} q} = \frac{x}{y} \tag{12}$$

- Requirement distribution (RD): the probability distribution used to partition the overall demand into individual order quantities. In this work, a normal distribution with a given average μ and given standard deviation σ is used. Thus, $RD = \mathcal{N}(\mu, \sigma^2)$.
- Overall capacity (OC): the ratio of the total volume of resources owned by two companies to that needed to meet the overall demand. Formally, $OC = x : y$ iff

$$\frac{c_1 + c_2}{\sum_{i=1,2} \sum_{q \in O_i} m_i * q} = \frac{x}{y} \tag{13}$$

- Resource distribution (ReD): the ratio of the volume of resources owned by one company to that of the other company. Formally, $ReD = x : y$ iff

$$\frac{c_1}{c_2} = \frac{x}{y} \tag{14}$$

4.1 Experimental Set-up

In our experiments, the following parameters are used: market price $d = 120$ and the number of resources required for production for each manufacturer $m_1 = m_2 = 4$. It is assumed that an agent requesting to purchase resources makes an initial offer of a payment of $b = 60\%$ of the marginal profits. If this offer is rejected, the agent increases the payment by $g = 20\%$ until it reaches 100% of marginal profits.

We have run five set of experiments whose input parameters are shown in Table 1. For each set of input parameters, 50 random scenario instances have been generated. The results shown are the averages over 50 experiments.

Set	Fixed Parameters	Variable Parameters
A	$DD = 1:9$, $RD = \mathcal{N}(10,1)$, $OC = 1:1$, $ReD = 9:1$	OD = 60, 70, 80, 90, 100, 110, 120, 130, 140, 150
B	$OD = 100$, $RD = \mathcal{N}(10,1)$, $OC = 1:1$, $ReD = 5:5$	$DD = 1:9, 2:8, 3:7, 4:6, 5:5, 6:4, 7:3, 8:2, 9:1$
C	$OD = 100$, $DD = 4:6$, $OC = 1:1$, $ReD = 5:5$	$RD = \mathcal{N}(5,1), \mathcal{N}(10,1), \mathcal{N}(15,1), \mathcal{N}(20,1), \mathcal{N}(25,1), \mathcal{N}(30,1)$
D	$OD = 100$, $DD = 4:6$, $RD = \mathcal{N}(10,1)$, $ReD = 5:5$	$OC = 1:10, 2:10, 3:10, 4:10, 5:10, 6:10, 7:10, 8:10, 9:10, 10:10, 11:10, 12:10$
E	$OD = 100$, $DD = 4:6$, $RD = \mathcal{N}(10,1)$, $OC = 1:1$	$ReD = 1:9, 2:8, 3:7, 4:6, 5:5, 6:4, 7:3, 8:2, 9:1$

Table 1. Experiment Settings

4.2 Results

The results of five experiments are shown in Table 2. In order to compare the performance of three scheduling approaches, the overall profit of LS and DS are presented as the percentage of GS. For each set of input parameters, the mean overall profit is displayed.

Table 2. Results of three scheduling approaches in experiments.

Set	Value	LS	DS	Value	LS	DS
A	60	0.142	1.000	110	0.199	1.000
	70	0.157	1.000	120	0.195	1.000
	80	0.192	1.000	130	0.193	1.000
	90	0.200	1.000	140	0.191	1.000
	100	0.200	1.000	150	0.192	1.000
B	1:9	0.599	1.000	6:4	0.898	1.000
	2:8	0.700	1.000	7:3	0.799	1.000
	3:7	0.799	1.000	8:2	0.698	1.000
	4:6	0.899	1.000	9:1	0.598	1.000
	5:5	1.000	1.000			
C	$\mathcal{N}$(5,1)	0.900	1.000	$\mathcal{N}$(20,1)	0.815	1.000
	$\mathcal{N}$(10,1)	0.899	1.000	$\mathcal{N}$(25,1)	0.874	1.000
	$\mathcal{N}$(15,1)	0.867	1.000	$\mathcal{N}$(30,1)	0.714	1.000
D	1:10	0.296	0.887	7:10	0.839	0.965
	2:10	0.458	0.988	8:10	0.990	1.000
	3:10	0.422	0.929	9:10	0.967	0.996
	4:10	0.982	0.999	10:10	0.924	0.999
	5:10	0.925	0.988	11:10	0.921	0.985
	6:10	0.859	0.995	12:10	0.990	1.000
E	1:9	0.698	1.000	6:4	0.799	1.000
	2:8	0.796	1.000	7:3	0.697	1.000
	3:7	0.895	1.000	8:2	0.599	1.000
	4:6	1.000	1.000	9:1	0.500	1.000
	5:5	0.897	1.000			

These results show that in our scenario and under our experimental conditions, the decentralised scheduling obtains largely globally optimal solutions solely by negotiating the price and quantity of resources. The substantially weaker total profit results for local scheduling show that genuine resource sharing benefits are being achieved in these scenarios. For example, Figure 2 illustrates this by plotting the overall profit achieved under LS and DS, relative to GS, under the experimental conditions of set A.

Figure 3 plots the overall profit of the LS and DS approaches as

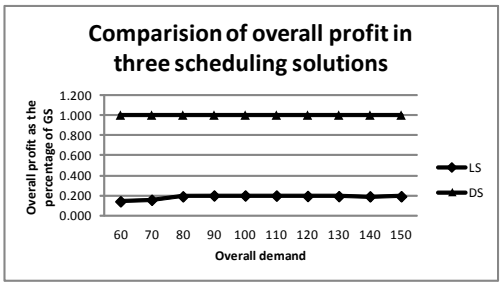

Figure 2. Results of experiment set A

the percentage of that of GS, for experiment set B. It shows that DS and LS generate the same profit result in scheduling instances with $DD = ReD$. Additionally, the bigger the difference of demand are distributed in two companies, the better DS works than LS when OD and ReD remain the same.

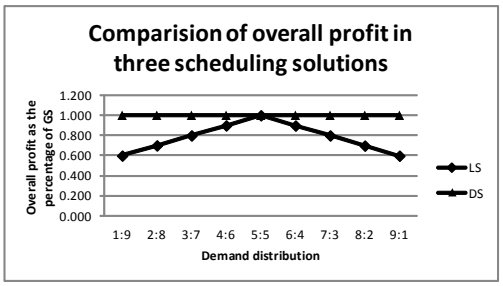

Figure 3. Results of experiment set B

There are circumstances where decentralised scheduling does not manage to produce a globally optimal solution as shown in Figure 4, which depicts the results of experiment set D. This is because in our approach the prospective purchaser initialises the resource request at the point with highest quantity and lowest payment. When the request is declined, it gradually increases the payment at first and then decreases the quantity when the payment reaches its upper bound.

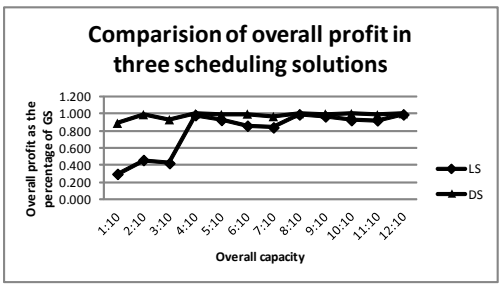

Figure 4. Results of experiment set D

5 RELATED WORK

An increasing number of papers examine developing agent-based inter-company scheduling methods in manufacturing. In them, the auction protocol has widely been used to guide agent negotiation behaviours [11]. Generally, the manager agent announces available resource capacities or production tasks and other agents reply with bids for prospective rewards [6].

Oliveira and Rocha [9] applied the auction protocol to develop an agent-based inter-company scheduling approach for scheduling production tasks as shown in Figure 5. In this approach, one customer task is not given to an individual manufacturer as private information, but published in an electronic market system. Moreover, internal scheduling decisions for production subtasks in each company should be submitted to the central market agent or delivery to other companies in order to resolve potential conflicts for processing the whole production task.

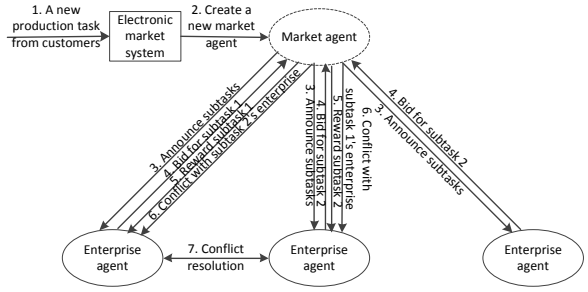

Figure 5. Example of an agent-based inter-company scheduling method

Some inter-company scheduling approaches allow individual companies to keep their production processes private, but require customer orders to be managed by third-party agents [2][13]. For example, the work presented in [8] introduces a yellow page agent maintaining the overall capacity of all involved manufacturers. The software agent common provider (SACP) representing a trusted third party is used in [1] to make scheduling decisions, which maintains a centralised repository of all available resources and a central scheduler to generate a schedule plan.

Another supply chain master planning approaches manage operational activities of organisational units for fulfilling customer demands [14]. They aim to enhance the overall competitiveness of the supply chain, especially to minimise the total relevant costs. Centralised methods may be only appropriate for static structures composed of isolated entities [10]. Although negotiation mechanism has been applied between buyer and supplier partners for collaborative planning, the private information is also exchanged (e.g. local cost effects of proposed resource supply [5]). This may be suited when both buyer and supplier come from the same organisation but not when they still compete with each other in the market.

6 CONCLUSIONS AND FUTURE WORK

This paper has presented a novel decentralised agent-based inter-company scheduling approach that operates through negotiation rather than sharing of potentially commercially sensitive information. This is useful in scenarios where the companies that need to coordinate their schedules are competitors and, therefore, reluctant to share explicit information on their orders, capacity and production plans with one another or with third party agents. We have examined the performance of our approach in a simple, two-company scenario. Through a set of experiments, we have shown that in this setting,

the benefits of resource sharing can be achieved and mostly globally optimal solutions are produced.

The work presented in this paper is a first step towards the development of fully decentralised inter-company scheduling approaches. As our approach is largely negotiation based, future work will elaborate the negotiation protocol and negotiation strategies. On the one hand, the use of richer communication language will allow the agents to negotiate more efficiently. This facilitates the discovery of globally optimal solutions. On the other hand, smarter negotiation strategies will limit the amount of (potentially commercially sensitive) information that is communicated to other agents. This prevents one agent from gaining insights into the privacy of the other. However, the agents' ability to discover globally optimal solutions may be limited as well.

Obviously, the scenario employed in this paper is a simple one. The scenario was kept simple with a view to be able to study it more effectively. Future work will examine the effect of such extensions. In particular, the introduction of a temporal dimension has many implications, not only for the simulation but the negotiation strategies as well. For instance, in such a setting, different types of resources (capital resources, resources that are consumed in production and perishable resources) raise different considerations. Also, the scheduling problem becomes a dynamic optimisation problem. Furthermore, in such a setting an agent's effectiveness is affected by its ability to predict future events (e.g. by learning from past experience).

Another important generalisation of the work is the introduction of additional agents. Again, this affects the negotiation in a variety of ways. In particular, adding agents makes negotiations multilateral instead of bilateral, which has implications on both the negotiation protocol and an individual agent's negotiation strategies. However, the computation complexity has to be considered when scaling up the inter-company scheduling problem.

REFERENCES

[1] A. Aerts, N. Szirbik, and J. Goossenaerts, 'A flexible, agent-based ICT architecture for virtual enterprises', *Computers in Industry*, **49**(3), 311–327, (2002).

[2] L. Camarinha-Matos, 'Multi-agent systems in virtual enterprises', in *Proceedings of the 11th International Conference on AI, Simulation and Planning in High Autonomy Systems*, (2002).

[3] L. Camarinha-Matos and H. Afsarmanesh, 'Virtual enterprise modeling and support infrastructures: Applying multi-agent system approaches', in *Multi-agent systems and applications*, eds., M. Luck, V. Marik, O. Stepankova, and R. Trappl, 335–364, Springer, (2001).

[4] P. Davidsson, S. Johansson, J. Persson, and F. Wernstedt, 'Agent-based approaches and classical optimization techniques for dynamic distributed resource allocation: A preliminary study', in *AAMAS03 workshop on Representations and Approaches for Time Critical Decentralized Resource/Role/Task Allocation*, (2003).

[5] G. Dudek and H. Stadtler, 'Negotiation-basedcollaborativeplanning between supply chains partners', *European Journal of Operational Research*, **163**(3), 668–687, (2005).

[6] S. Kraus, 'Automated negotiation and decision making in multiagent environments', in *Multiagent Systems and Applications*, eds., M. Luck, V. Marik, O. Stepankova, and R. Trappl, 150–172, Springer, (2001).

[7] L. Monostori, J. Vancza, and S. Kumara, 'Agent-based systems for manufacturing', *CIRP Annals - Manufacturing Technology*, **55**(2), 697–720, (2006).

[8] T. Norman, A. Preece, Chalmers. S., N. Jennings, M. Luck, V. Dang, T. Nguyen, V. Deora, J. Shao, W. Gray, and N. Fiddian, 'Agent-based formation of virtual organisations', *Knowledge-Based Systems*, **17**(2-4), 103 – 111, (2004).

[9] E. Oliveira and A. Rocha, 'Agents advanced features for negotiation in electronic commerce and virtual organisations formation processes', in *Agent Mediated Electronic Commerce*, eds., F. Dignum and C. Sierra, 78–97, Springer, (2001).

[10] R. Pibernik and E. Sucky, 'An approach to inter-domainmasterplanning in supplychains', *International Journal of Production Economics*, **108**(1-2), 200–212, (2007).

[11] J. Reaidy, P. Massotte, and D. Diep, 'Comparison of negotiation protocols in dynamic agent-based manufacturing systems', *International Journal of Production Economics*, **99**(1-2), 117–130, (2006).

[12] W. Shen, Q. Hao, H. Yoon, and D. Norrie, 'Applications of agent-based systems in intelligent manufacturing: An updated review', *Advanced Engineering Informatics*, **20**(4), 415–431, (2006).

[13] W. Shen, F. Maturana, and D. Norrie, 'MetaMorph II: An agent-based architecture for distributed intelligent design and manufacturing', *Jounal of Intelligent Manufacturing*, **11**(3), 237–251, (2000).

[14] H. Stadtler, 'Supply chain management and advanced planning-basics, overview and challenges', *European Journal of Operational Research*, **163**(3), 575–588, (2005).

[15] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 2000.

[16] M. Wooldridge, *An introduction to multiagent systems*, Wiley, 2009.

Improving Production Scheduling with Machine Learning

Jens Heger[1], **Hatem Bani**[1], **Bernd Scholz-Reiter**[1]

Abstract. Decentralized scheduling with dispatching rules is applied in many fields of production and logistics, especially in highly complex manufacturing systems, e.g., semiconductor manufacturing. Nevertheless, no dispatching rule outperforms other rules across various objectives, scenarios and system conditions. In this paper we present an approach to dynamically switch dispatching rules depending on the current system conditions. The rules' performances are estimated by machine learning methods, whose learning data is generated by preliminary simulation runs. A comparison between neural networks and Gaussian processes as regression methods is conducted. Using a dynamic job shop scenario we demonstrate, that our general approach is capable of significantly reducing the mean tardiness of jobs; with Gaussian processes leading to better results.

1 INTRODUCTION

In today's highly competitive, globalized markets, manufacturing companies have to use their production resources as efficiently as possible. Therefore, especially capital-intensive industries like semi-conductor manufacturing spend considerable effort to optimize their production processes. Improvements in scheduling lead to a better achievement of objectives (e.g., tardiness of jobs). Scheduling in job shops or flexible flow shops is a combinatorial, NP-hard optimization problem. These problems have attracted researchers and practitioners for many decades now and are still of considerable interest, because of their high relevance. Many heuristics, which calculate schedules in a centralized manner, have been introduced, since optimal solutions can only be calculated for small scenarios. If the production scenarios are facing high variability like continuously arriving new jobs, job changes, break-downs etc. decentralized scheduling methods are advantageous compared to central methods. One class of decentralized scheduling heuristics are dispatching rules ([1], [2]), which are widely used to schedule even very complex shop floors. Their popularity derives from the fact that they perform reasonably well in a wide range of environments, and they are relatively easy to understand. Furthermore, they only require minimal computational time, which qualifies them to be used in real-time, online scheduling. Therefore, they can always take the latest information available from the shop-floor into account. Many dispatching rules are proposed in the literature, which perform well on specific scenarios. However, no rule is known to consistently outperform all other rules [3]. One approach to meet this challenge and improve scheduling performance is to select and switch dispatching rules depending on current system conditions. For this task machine learning methods, e.g. artificial neural networks [4], are frequently used.

In this paper we present a comparison between artificial neural networks [5] and Gaussian processes ([6], [7]) as learning methods to determine which method performs better estimating dispatching rule performance. We also investigated how the number of learning data points affects the quality of the learned models for our chosen scenario. This paper is organized as following: In section 2 we give a review of previous work on dispatching rules, machine learning in scheduling and introduce Gaussian processes and artificial neural networks. In section 3 our chosen scenario and the experimental designs are described. Section 4 presents the results of our experiments. The paper concludes with a short summary and provides directions towards future research.

2 STATE OF THE ART

2.1 Scheduling with dispatching rules

Scheduling is defined by Haupt [1] as "the determination of the order in which a set of jobs (tasks) {i | i = 1, ..., n) is to be processed through a set of machines (processors, work stations) (k | k=1...m)." Since the problem is NP-hard, heuristics are used. Especially in extremely complex scenarios with high variability dispatching rules are often employed. Dispatching rules are applied to assign a job to a machine. This is done each time the machine becomes idle and there are jobs waiting. The dispatching rule assigns a priority to each job. This priority can be based on attributes of the job, the machines or the system. The job with the highest priority is chosen to be processed next. Dispatching rules have been developed and analyzed in the scientific literature for many years; see e.g. [1], [2] and [8]. The best known rules are Shortest

[1] BIBA - Bremer Institut für Produktion und Logistik GmbH at the University of Bremen, Hochschulring 20, 28359 Bremen, Germany, Tel.: +49 421 218-50103, {**heg**|ban|bsr}@biba.uni-bremen.de

Processing Time first (SPT), Earliest Due Date (EDD) and First Come First Served (FCFS).

2.2 Machine learning in Scheduling

Kotsiantis [11] gives an overview of a few supervised machine learning techniques, like artificial neural networks, decision trees, Naïve Bayes, support vector machines etc. Priore et al. [12] present a review of machine learning in dynamic scheduling of flexible manufacturing systems. Most approaches are based on artificial neural networks and are described in the following.

A simulation-based approach was presented by Wu and Wysk [13]. They switch regularly between different dispatching rules on machines. They proposed a multi-pass scheduling algorithm, which starts a short-term simulation of alternative rules and selects the best candidate for the manufacturing system.

A neural network based controller, consisting of an adjustment module and the equipment level controllers, was proposed for scheduling and controlling a manufacturing cell by Sun and Yih [14]. The adjustment module considers the user objectives and the current performance levels to determine the relative importance of performance measures. Based on these importance values and current machine status, the equipment level controller, implemented by a neural network, selects a proper dispatching rule and the jobs are processed accordingly. The training samples for each equipment level controller are calculated by a one-machine simulation and modified to reflect the impacts of different dispatching rules on the system performance.

El-Bouri et al. [15] used a neural network to select dispatching rule in a job shop. They chose small scenarios with five machines and investigated three rules. To train the neural network they calculated optimal solutions for 10, 15 and 20 jobs. The neural network was used to select one rule for every machine. With this approach they were able to get better results than just using one of the rules on every machine. The drawback of this approach is that it is limited to scenarios with only a few machines and jobs, otherwise no optimal solutions for learning could be generated.

Mouelhi-Chibani and Pierreval [4] use a neural network to dynamically switch dispatching rules on every machine depending on the current system state. They have selected four system parameters (e.g. shop load) and 22 system state variables (e.g. average slack time of jobs in the first queue), which the neural network uses to decide which rule should be applied. They train the neural network with preliminary simulation runs. The scenario they selected consists of only two machines and the set of dispatching rules consists of SPT and EDD. They outperform the static use of rules, but not that clearly, which might be due to the small scenario.

These are interesting approaches, but the results seem to have potential for improvement. It is not clear if this is due to the selected scenario or the learning technique.

In [16] we have conducted a first study how Gaussian processes perform in the application of dispatching rule switching. A preliminary comparison with other learning techniques, e.g. artificial neural networks has been performed.

In this paper a more detailed and solid comparison between both methods including the optimization of parameter settings and an evaluation in a dynamic simulation study is conducted.

2.3 Machine learning methods

Alpaydin [17] stated: “The goal of machine learning is to program computers to use example data or experience to solve a given problem”. A common choice as a machine learning method are artificial neural networks. A relatively new and promising method is Gaussian process regression. In this study, we are interested in a system that can predict the value of an objective function from production system characteristics, which otherwise would have to be obtained by costly simulations. To analyze which of these methods suits better for our field of application, we compare them in this study.

2.3.1 Neural Networks

Artificial Neural Networks have been studied for decades and Hornik [18] has shown that “…standard feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any degree of accuracy, provided sufficiently many hidden units are available.”

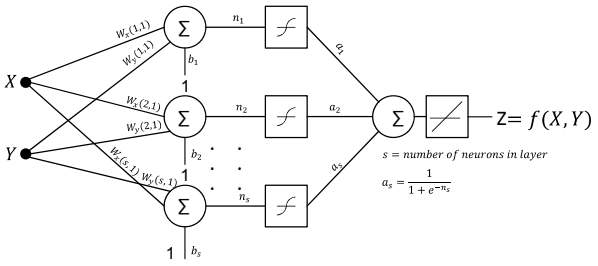

Figure 1: Neural network architecture with one hidden layer

Geva and Sitte [19] additionally demonstrate, that a feedforward multilayered neural network, based on neurons with sigmoidal transfer function, is able to approximate arbitrarily well any continuous multivariate function. In many engineering research areas function approximation is used to improve processes. Multilayer feedforward neural networks with sigmoidal activation functions are a powerful tool to approximate these functions [5]. Figure 1 shows the architecture of a multilayer feedforward neural network with one hidden layer and the sigmoid transfer function.

2.3.2 Gaussian Processes

O'Hagan [20] represents an early reference from the statistics community for the use of a Gaussian processes as a prior over functions, an idea which was introduced to the machine learning

community by Williams et al. [7]. As stated before we have a simulation model implicitly implementing a (noisy) mapping between a vector of state variable (in our case containing, e.g. utilization) and the objective function (mean tardiness) $y = f(x) + \varepsilon$. The learning consists of finding a good approximation f*(x) of f(x) to make predictions at new points x. To learn such a model using Gaussian processes requires some learning data as well as a so-called covariance function. This covariance function, sometimes called kernel, specifies the covariance between pairs of random variables and influences the possible form of the function f* learned. A common choice for the covariance function is the squared exponential (SE) covariance. It is depicted in equation (1):

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq} \qquad (1)$$

The squared exponential covariance function has three hyperparameters. There is the length-scale *l*, the signal variance σ_f^2 and the noise variance σ_n^2 with δ_{pq} being a Kronecker delta function, which is 1 if p=q and zero otherwise. These parameters of a covariance function can be used to fine-tune the GP-model, thus learning of a Gaussian processes model requires having some learning data, choosing an appropriate covariance function and choosing a good set of hyperparameters (see ([6] chapters 2 and 4).

To learn, or optimize the hyperparameters, the marginal likelihood should be maximized. Details and mathematical background can be found in ([17] chapter 5), especially equation (5.9) page 114. Basically, the hyperparameters are chosen in a way that the generalization error, which is the average error on unseen test examples, is minimized. This is done with cross-evaluation by splitting the training data in learning and test data. The training error is not optimized, because this may lead to over-fitting the data. Additionally, since hyperparameters can be interpreted as length-scale parameters in the case of the squared exponential covariance, additional optimizations can be performed. Rasmussen and Williams [6] describe the hyperparameters informally like this: "…how far do you need to move (along a particular axis) in input space for the function values to become uncorrelated…". Thus, the squared exponential covariance function implements automatic relevance determination (ARD) [21] since the inverse of the length-scale determines how relevant an input is. A very large length-scale value means that the covariance will become almost independent of that input. ARD has been used for removing irrelevant input by several authors, e.g. [6].

3 APPROACH & EXPERIMENTAL SETUP

The main focus of our research is to develop a new scheduling method, which uses local and global information to make scheduling decisions. Therefore, we suggest learning performance models of the dispatching rules with a machine learning method to select the best rule for the current conditions. This is a promising approach, since the major drawback of dispatching rules is their lack of a global view of the problem. Rules approach the overall scheduling problem by taking independent scheduling decisions based on the current, local conditions at the particular machine; without consideration of the negative effects they might have on future decisions and on the overall objective function value.

The performance models are learned by preliminary simulation runs, which add a global perspective on the system behavior to the local decision rules. Here we investigate if Gaussian processes or artificial neural networks perform better in our field of application.

3.1 Scenario description

The type of problems we address, are dynamic shop scenarios. Our computational experiments are based on the dynamic job-shop scenarios from Rajendran and Holthaus [3]. In total there are 10 machines on the shop floor, each job entering the system has to visit each machine once, using a random routing, i.e., machine visitation order is random with no machine being revisited. Processing times are drawn from a uniform discrete distribution ranging from 1 to 49 minutes. The due dates of the jobs are determined by a due date tightness factor, a job's due date is set to x-times the job's total processing time + release time. Job arrival is a Poisson process, i.e., inter-arrival times of jobs follow an exponential distribution. The arrival rate is set to yield a desired long term utilization level of each machine.

The dynamic experiments simulate the system for a duration of 12 months, using changing utilization rates and due date factors. All results in section 4.3 are based on these dynamic settings. The utilization rate of the shop, i.e., arrival rate, is oscillating between 0.75 and 0.99 following a sine function with a period length of 30 days. The sine function to generate due date factors has a period length of 15 days and oscillates between 2 and 7. Performance figures are calculated averaging the tardiness of all jobs started within the simulation length of 12 month. To generate the learning data we are only interested in the performance for a specific setting of utilization and due date tightness. Therefore, we closely follow the procedure from Rajendran and Holthaus [3]. We start with an empty shop and simulate the system until we collected data from jobs numbering from 501 to 2500. The shop is further loaded with jobs, until the completion of these 2000 jobs [8]. Data on the first 500 jobs is disregarded to focus on the shop's steady state behavior.

3.2 Switching dispatching rules

We have selected two dispatching rules, out of which the best for each system condition can be selected. The first is a standard rule being used for decades; the second rule was developed by Holthaus and Rajendran [22] especially for their scenarios. If the rules calculate the same priority for more than one job, we use ERD (*earliest release date*) as a tiebreaker. Our approach works with more than two rules, but the selected two outperform standard rules from the literature in most cases.

MOD –*Modified Operation Due Date*: MOD orders the queue of waiting jobs by the larger of each job's operation due date ($d_{i,imt}$) minus the current time (t) or each job's operation processing time ($p_{i,imt}$). Therefore, if all jobs in the queue have positive slack (no job is in danger of missing its due date) then MOD dispatches them in earliest operational due-date (ODD) order. If all jobs have negative slack (all jobs are in danger of missing their due dates), then MOD works like SPT to reduce shop congestion. Definition:

$$MOD_i = \max(p_{i,imt}, d_{i,imt} - t) \quad (2)$$

2PTPlusWINQPlusNPT – 2Processing Time + Work in Next Queue + Next Processing Time: This rule [22] consists of three parts. First, the processing time on the current machine is considered. Secondly, the Work in Next Queue is added: WINQ – jobs are ranked in the order of a estimation of their waiting time before processing on the next machine can start. This estimation includes the time needed by a machine m to finish its current job plus the sum of processing times of all jobs currently waiting in front of *m*. The job where this sum is least has the highest priority. Thirdly, the processing time of a job's next operation NPT is added. Definition:

$$2PTPlusWINQPlusNPT_i = 2p_{i,imt} + WINQ_i + p_{i,imt+1} \quad (3)$$

3.3 System architecture

For the simulation experiments of this paper we use Jasima [24], a self-implemented discrete-event simulation. Jasima is very roughly based on a Java-port of the SIMLIB library [9] (described in [10]). For the Gaussian processes, we have used the software examples provided by Williams [23] and adapted them for our scenarios. They have been implemented with MatLab from MathWorks. For the implementation of the artificial neural networks we have used the Neural Network Toolbox from MatLab. The MatLab Builder JA is used as an interface between the simulation software and the Gaussian processes and the neural networks respectively.

4 EXPERIMENTS AND RESULTS

4.1 Machine learning settings

4.1.1 Selection of learning data

To learn performance models of the two selected dispatching rules, we performed preliminary simulations runs with both rules and different system conditions. For the system conditions we selected two parameters, which are the input for the machine learning methods. The first is the system's utilization and the second is the due date factor, which defines the job's due date tightness. These two system parameters have been combined in 1525 combinations. We have performed simulation runs with system utilizations from 75% till 99% and have combined each of these with due date factors from 1 to 7 (in 0.1 steps). The two selected dispatching rules described in section 3.1 have been evaluated for all these parameter combinations. Our performance criterion is mean tardiness, but the general approach is applicable to other objective functions as well. Each result for each combination of utilization, due date factor and dispatching rule is the average of 20 independent replications to get reliable estimates of the performance of our stochastic simulation (see figure 2).

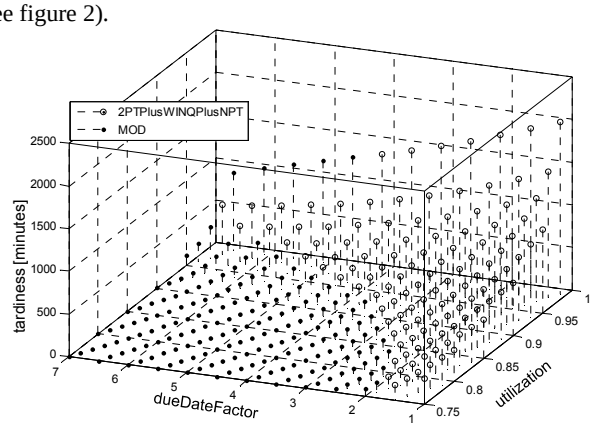

Figure 2. Results of preliminary simulation runs with 1525 parameter combinations (for better clarity some have been omitted; only best performing rule shown).

If simulation runs are expensive and more system parameters are considered, not all parameter combinations can be simulated in advance. This is where supervised machine learning techniques can play an important role, helping to select the best dispatching rule with only a few simulations runs as a learning data set. Therefore, we also investigated how the number of learning data points affects the quality of the learned models and the scheduling results in the end for our chosen scenario. For the selection of data points, i.e. a combination of utilization rate and due date factor, we used 500 latin hypercube designs (c.f. [25]) with set sizes of 10, 15, 20, 30, 45, 60, 75, 120 and 350 data points each.

4.1.2 Artificial Neural Networks

For our study we have chosen a feedforward multilayered neural network. As a training algorithm the Marquardt algorithm has been used. It is incorporated into the backpropagation algorithm, which has been tested on several function approximation problems [26]. As an regularization technique we used early stopping [27].

Another important parameter to set is the number of hidden neurons. Geva and Sitte claim that it is not some arbitrary number, but it should be rather set proportional to the number of function points in the training set [19]. Pei et al. show that a neural network can be used as an 'universal approximator', but the number of hidden neurons cannot be generally calculated, thus it imposes a signifi-

cant practical challenge [5], [28]. Therefore, we performed a preliminary study (see table 1), to determine which number of neurons leads to best results depending on the number of learning data in our field of application and use these later on.

Table 1. Best number of neurons for different training set sizes and resulting decision errors

	Number of neurons					
Training set size	2	5	10	20	30	50
NN 10	**283.2**	283.9	238.3	299.2	330.6	415.1
NN 15	227.3	**166.8**	195.0	197.9	245.1	347.3
NN 20	127.3	**118.0**	153.6	174.2	211.0	296.0
NN 30	85.0	**82.8**	103.9	144.9	161.2	258.9
NN 45	75.0	**48.9**	59.8	98.4	116.8	168.2
NN 60	51.9	41.8	**41.4**	53.7	99.1	151.4
NN 75	54.4	27.0	**25.8**	29.4	60.7	98.9
NN 120	38.6	18.9	**16.9**	23.5	20.3	61.7
NN 350	25.4	11.5	6.6	**2.5**	3.6	3.7

4.1.3 Gaussian Process regression

We use the squared exponential covariance with automatic relevance detection and white noise for our analysis. As a mean function we used the sum of a linear and constant function with start setting 0.0 [6]. Further we have investigated the start settings for the hyperparameters with some example data. Noise σ_n^2 has been set to values between log(0.01) for a small number of learning data points and log (0.1) for many learning points. Lengthscale factors have been initialized with 0.1 and 2.5 and the signal variance σ_f^2 has been set to 1000.

4.2 Comparison of Gaussian Processes and Artificial Neural Networks

For our experiments we have used 500 different sets for each number of learning points and calculated a decision error for each model. The error is calculated by summing up the wrong decisions of each model for each possible combination. The error is the difference between the best and the selected rule, e.g. if we have to test the parameter combination 0.83 utilization and due date factor 3, the model gives us an estimates of 150 minutes for MOD, and 180 for 2PTPlusWINQPlusNPT. So we would select MOD. If the true values are 200 for MOD and 175 for 2PTPlusWINQPlusNPT the error would be 25 minutes. Figure 3 shows the results of our study and it can be seen, that the Gaussian processes outperform the Neural networks significantly (twice standard error) for all numbers of data points.

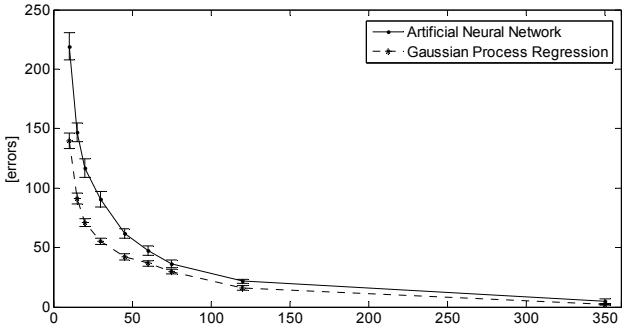

Figure 3: Results of 1525 tested parameter combinations for 500 different data point set for each number of learning data (twice standard error shown)

4.3 Simulation experiments

In addition to the static analysis we have conducted a simulation study, to evaluate our results in a typical dynamic shop scenario (see section 3.2). We have selected 50 different training sets, generated with latin hypercube designs, with 30 data points each. 30 data points is a good compromise between the already well learned models and the number of needed simulation runs.

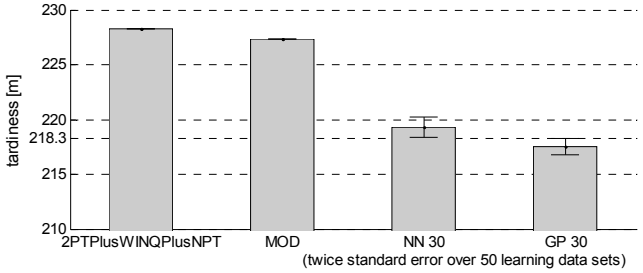

Figure. 4. Simulation results of the dynamic scenario. Two standard rules compared with the performance of switching rules based on neural network and Gaussian process models with 30 learn data points in 50 different sets

The results in and figure 4 and table 2 show that Gaussian processes outperform artificial networks significantly (twice standard error) in this dynamic scenario, which confirms our static analysis.

Table 2. Dynamic scenario simulation results

Rules	**tardiness [m]**	**Twice standard error (50 learning sets)**
2PTPlusWINQPlusNPT	228.3	
MOD	227.3	
NN 30	219.3	(0.97)
GP 30	217.5	(0.74)

The results of the dynamic simulation study also show, that scheduling with dispatching rules can be improved by >4% with only 30 learning data points, i.e., preliminary simulation runs.

5 CONCLUSION AND NEXT STEPS

In dynamic manufacturing scenarios with frequently changing system parameters, adaptive scheduling approaches improve the performance of dispatching rule based scheduling. In our study we have compared the performance of artificial neural networks with Gaussian process regression in learning dispatching rule behavior under different system conditions. In our static analysis we have shown that Gaussian processes perform significantly better than neural networks regardless of how many data points are used. We have confirmed our findings in a dynamic simulation study, where Gaussian processes also perform significantly better.

The scheduling performance compared to standard dispatching rules can be improved by over 4% in our chosen scenario. In further studies the underlying scenario could be extended e.g. to semiconductor manufacturing, which is more complicated (i.e. sequence depending setup times, batch machines etc.). Switching rules in such a scenario might increase the performance even more, e.g. when the product mix changes and a batch machine becomes the bottleneck, the effect of different rules on the objective can be severe. Additionally, simulation costs increases, which makes a good selection of learning data more important.

ACKNOWLEDGEMENTS

The authors are grateful to the generous support by the German Research Foundation (DFG), grant SCHO 540/17-2.

REFERENCES

[1] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *International Journal of Production Research*, 20(1):27–45, 1982.

[2] R. Haupt, "A survey of priority rule-based scheduling," *OR Spektrum*, 11(1):3–16, 1989.

[3] C. Rajendran and O. Holthaus, "A comparative study of dispatching rules in dynamic flowshops and jobshops," *European Journal of Operational Research*, 116(1):156–170, 1999.

[4] W. Mouelhi-Chibani and H. Pierreval, "Training a neural network to select dispatching rules in real time," *Computers & Industrial Engineering*, 58(2):249 – 256, 2010, scheduling in Healthcare and Industrial Systems.

[5] Jin-Song Pei, Joseph P. Wright, and Andrew W. Smyth. Mapping polynomial fitting into feedforward neural networks for modeling nonlinear dynamic systems and beyond. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4481 – 4505, 2005.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[7] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems*, 8:514–520, 1996.

[8] S. S. Panwalkar and W. Iskander. A survey of scheduling rules. *Operations Research*, 25(1):45–61, 1977.

[9] A. M. Law. *Simulation Modeling and Analysis*. McGraw-Hill, New York, USA, 4th edition, 2007.

[10] B. J. Huffman. An object-oriented version of SIMLIB (a simple simulation package). *INFORMS Transactions on Education*, 2(1):1–15, 2001.

[11] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[12] Paolo Priore, David de la Fuente, Alberto Gomez, and Javier Puente. A review of machine learning in dynamic scheduling of flexible manufacturing systems. *AI EDAM*, 15(03):251–263, 2001.

[13] Szu-Yung David Wu and Richard A. Wysk. An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *International Journal of Production Research*, 27(9):1603–1623, 1989.

[14] Y. L. Sun and Y. Yih. An intelligent controller for manufacturing cells. *International Journal of Production Research*, 34(8):2353–2373, 1996.

[15] Ahmed El-Bouri and Pramit Shah. A neural network for dispatching rule selection in a job shop. *The International Journal of Advanced Manufacturing Technology*, 31(3-4):342–349, 2006.

[16] B. Scholz-Reiter, J. Heger, and T. Hildebrandt, "Gaussian processes for dispatching rule selection in production scheduling," *Procceeding of the International Workshop on Data Mining Application in Government and Industry 2010 (DMAGI10) As Part of The 10th IEEE International Conference on Data Mining*, pp. 631–638, 2010.

[17] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning Series)*, vol. 14. MIT Press, 2004.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[19] S. Geva and J. Sitte. A constructive method for multivariate function approximation by multilayer perceptrons. *Neural Networks, IEEE Transactions on*, 3(4):621 –624, 1992.

[20] A. O'Hagan. Curve fitting and optimal design. *Journal of the Royal Statistical Society*, 40(1):1–42, 1978.

[21] Radford M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, August 1996.

[22] O. Holthaus and C. Rajendran. Efficient jobshop dispatching rules: further developments. *Production Planning & Control*, 11(2):171–178, 2000.

[23] Christopher K. I. Williams. Gaussian processes for machine learning software examples. gaussianprocess.org/gpml/code/matlab/doc, 2006.

[24] http://code.google.com/p/jasima/

[25] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):pp. 239–245, 1979.

[26] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989 –993, 1994.

[27] S. Amari. Training error, generalization error and learning curves in neural learning. In *Proceedings of the 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems*, ANNES '95, pages 4–, Washington, DC, USA, 1995. IEEE Computer Society.

[28] D. Kriesel. *Ein kleiner Überblick über Neuronale Netze*. 2007. http://www.dkriesel.com

Hypotheses Generation for Process Recognition in a Domain Specified by Temporal Logic

Immo Colonius [1]

Abstract. The degree of automation in the logistic domain is increasing, which amplifies the need of autonomous robotic systems. An autonomous system requires situational awareness as well as an understanding of processes in its environment to act goal orientated. Process Recognition methods show valid solutions for this problem. Considering the challenge of recognizing processes from a humans point of view, explainability of found solutions and simple process definitions for processes to search for becomes essential. This leads to the use of qualitative reasoning techniques. Especially logistic domains like warehouses are highly dynamic and mobile robots situated in that domain often do not have complete sensor coverage of the whole environment. Logic methods like deduction for model checking can thus be insufficient due to incomplete or wrong observations, breaking the reasoning chain. Topic of this paper is the enhancement of the deductive reasoning for process recognition with hypotheses based on background knowledge and reasoning techniques to overcome these limitations. Therefore, a qualitative measurement for certainty will be motivated and benefits of hypotheses integration shown.

1 INTRODUCTION

This paper is a follow up of a previous paper that shows the use of logic process recognition by Linear Temporal Logic (LTL) in the logistic domain [19]. There are two new features: Reasoning about all histories of goods simultaneously, which enables the use of inference knowledge at the time of reasoning and more important, enhancement of the reasoning process with hypothetical facts. In this paper, a hypothetical fact is a virtual observation inserted in our knowledge base to close gaps in actual observations. The later enhancement is overcoming the previous limitations of the model checking based approach of not being able to find processes if a crucial observation is missing. Therefore, we introduce a qualitative uncertainty description that works well with the present logic reasoner.

Understanding processes happening in an autonomous agents environment is essential for goal orientated action. Especially in dynamic environments, known as a demanding challenge that involves recognition as well as understanding of processes for the agent. An example is automated construction of simulations for logistic domain optimization [13, 1], like flow simulations for goods in warehouses or productions lines. This challenge demands an robotic observer, which is capable of paying attention to logistic processes around him while avoiding dangerous zones (staying out of redistribution processes or hazardous machines). Additionally scenarios alike offer limited observability for a mobile, non intrusive robot, which makes process recognition non trivial, as not all information is provided or may even be misinterpreted due to sensor noise.

With the demand of safety, plans verification for the control of the mobile robot is desired, which leads to the use of formal logic [22, 11, 4]. Deployment of logic for process recognition yields other advantages, too: Explainability of recognized processes is easier human understandable than a trained process recognizer like a Neural Network [17] or Markov Decision Process models [5]. Also system setup, i.e. input of process definitions for model checking, of processes that can be found in the given domain (like an admission of a good in a warehouse) can be made possible for domain experts without knowledgeability of robotic systems. Logics further provide a flexible basis for late changes to queries without the need to relearn a scenario. Additionally, by working on abstracted facts the reasoning process is more portable than a domain-trained system.

The use of logic for robot control is widespread, specification of controllers by a correct by construction method is introduced by Kress-Gazit [18]. Motion planning from high level specifications is common [15, 25, 20, e.g.] and Kloetzer [16] demonstrate the applicability to real robotic systems.

The use of logic reasoning for process recognition in the logistic domain has been shown in [19] on real world data from an model warehouse, even handling partial observability by the robotic observer to a degree. Coping with incomplete data still remains a difficulty on the logic reasoning level, even more difficulties arise when wrong observations are incorrectly abstracted and result in contradictory facts.

This paper argues for the use of hypotheses generation techniques and incorporation of hypothetical facts as well as a qualitative measurement of certainty into the reasoning process to narrow the gaps in complete environment understanding left open by partial or incorrect observability of the domain.

2 RELATED WORK

A multitude of different approaches exists for the problem of process recognition and they can be categorized into learning approaches, probabilistic process descriptions, and logic-based declarative approaches.

Examples for the learning approaches are Markov networks [3, 21], Bayesian networks [27], supervised learning [2], or inductive logic programming [7], which require a training phase before deployment. As written in [19], we fancy an approach that does not need a training phase. This enables us to pose dynamic queries to our knowledge base in a flexible formal language and to integrate new knowledge at any point of the experiment.

Possible solutions are declarative, logic-based formalisms [11, 4].

[1] University of Bremen, Germany, email: colonius@informatik.uni-bremen.de

They are already in use, for example integration of ontology based knowledge to recognize contexts in an ubiquitous environment by Mastrogiovanni [23] or in controllers construction for robot control.

Combinations for logic and uncertainty are also present; In contrast to Fagin [10] we try to avoid a metric basis for our qualitative notes altogether. While Ilić-Stepić [14] uses qualitative probability to compare likelihood of facts against each other, we use the notion of uncertainty to rate facts and conclusions. Another approach was done by Goldszmidt [12], who uses qualitative uncertainty to model if/then clauses. For a well done overview about differences between descriptions of uncertainty in possibilities, probability theory and multi-valued logics see [8].

In our process recognition approach, we match observations against pre-proposed processes, which is essentially a model checking approach. Model checking is widely known in software verification, as well as robotics. Planning instances in this field are found in [6, 9, 16, e.g.].

3 PROBLEM DESCRIPTION

3.1 PROCESS RECOGNITION

Imagine we have a description of processes that happen in our domain. Now, we have several observations about the state of our environment. Our goal is an understanding what exactly has happened in the environment, so to speak finding instances of our processes, based on an assemble of our observations. Also, take into consideration that our environment is not fully observable and our observations may not be entirely correct. This presents us with several problems, additional to common process recognition, to solve:

- Uncertainty of made observations:
  If we can't be sure that an observation is true, we may conclude wrong processes. Which leads to a general need for abstraction of facts in order to lessen the impact of wrong observations.
- Observability of our domain:
  If we can't observe critical elements of processes happening, we can't infer named processes by pure deductive reasoning. That leads to guesswork to achieve maximal environment understanding. We have to close the gaps in our observations by somewhat sensible hypotheses.

Our approach yields a high degree of generality. As our domain can be highly dynamic, means to lessen the degree of noise from sensor measurements, like done with abstraction towards logic facts are desired.

A possible scenario is sketched in the following subsection.

Example for the Warehouse Scenario

Our process recognition problem is placed in so-called in *chaotic* or *random-storage warehouses*. Their main features are absence of a central control structure which often leads to incomplete knowledge about the state of the warehouse. Especially if more than one entity moves goods around. Additionally, a steady flow of moving goods through space is present. The observers task is to recognize the storage processes that occur. However, as a observer is generally not able to gather all potentially relevant information about a process and therefore needs to infer missing pieces of information, in particular identifying functional locations. Although more complex processes, such as inference about unnecessary redistributions or not

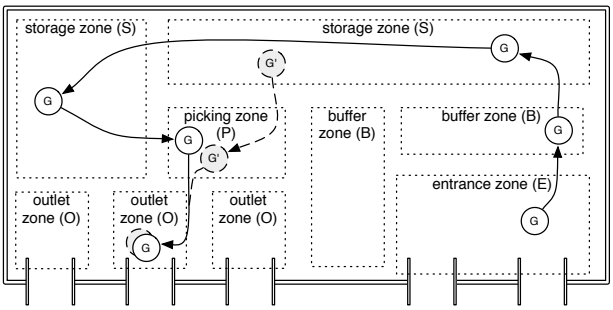

Figure 1. A warehouse, its functional zones, and typical movements of different goods (G, G')

needed spaces can be done, this paper presents a shorter one with respect to limited space.

An example of stockpiling of a good can consist of:

- Bringing a good into the warehouse (making an observation $obs(G, L, T)$ of a good G at a location L at time T that has not been seen so far).
- Placement of the same good G at another location L_{+n} at a later point of time (making an observation later on T_{+n} at another place L_{+n} than the one were the good G has been seen before). (here n denotes a change and m a greater change)
- Second placement at another place even later on (making an observation even later on T_{+m} at a third location L_{+m}).

Storage processes are defined by a unique pattern [26]: on their way into and out of the warehouse, goods are (temporarily) placed into *functional zones* which serve specific purposes (see Fig. 1). All goods arrive in an *entrance zone* (E). From there, they are picked up and temporarily moved to a *buffer zone* (B) before they are finally stored in the *storage zone* (S). This process is called 'admission'. Within the *storage zone* 'redistribution' of goods can occur arbitrarily. When 'taking out' goods, they are first moved from the *storage zone* to the *picking zone* (P) from where they are taken to an *outlet zone* (O), before being moved out of the warehouse. Note that functional zones can contain any number of locations and have a dimensional extend, while locations are points where goods have been observed.

A working approach using logic deduction for process inference has been shown in a recent publication [19], where this example originates. Incomplete observed processes were not successfully inferred as no hypotheses generation has been deployed, so that the quality of the results were directly proportional to sensor coverage and abstraction correctness.

Our task is to find suitable means to insert sensible hypotheses to cover missing or wrong observations. As our observations are bound to a spatial and temporal context, hypothetical observations have to constructed likewise.

3.2 Example for Deductive Model Checking

Recognition of a process by model checking is done by fulfilling all conditions of a process model (e.g. $Admission$-formalization in Fig. admission-example-2) by found instances (e.g. at() primitives). Used on our warehouse scenario, we have a good G, which enters the warehouse and is stored in the entrance zone E at position L_1 at time t_0. Movements occur between t_1 and t_2 (the good is moved to a location L_2) and between t_2 and t_3 (the good is moved further to

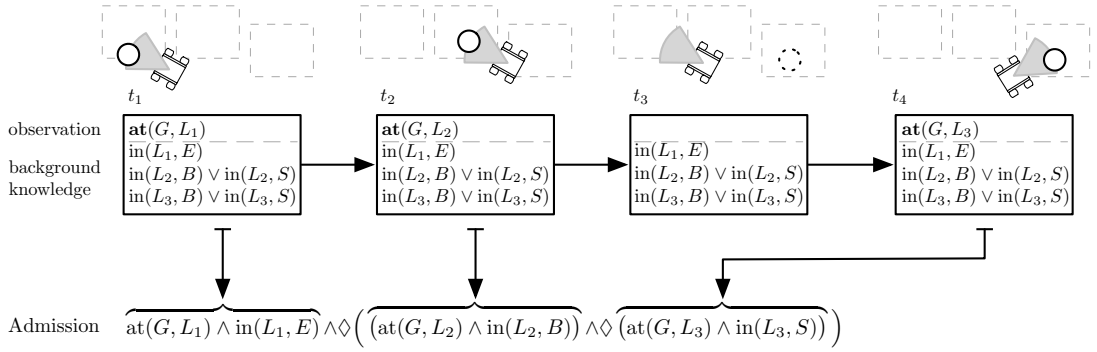

Figure 2. Example: Model checking for an admission process of good G (only the relevant assertions for each world $t_{1\ldots4}$ are shown). $\mathrm{in}(L_1, E)$ is background knowledge, also it is known that locations L_2 and L_3 are either part of the buffer zone (B) or the storage zone (S) but not close to one another so that they do not have to belong to the same zone. From this admission refined knowledge about the buffer and storage zones can be inferred: $\mathrm{in}(L_2, B) \wedge \mathrm{in}(L_3, S)$ [19].

L_3). Our mobile robot records the following observations: G to be at L_1 at t_1, G to be at L_2 at t_2 and G to be at L_3 at t_4 .

Fig. 2 displays an example for an (1), i.e., the observed process is an admission that starts in world t_1 and ends in world t_4. Our zone inference reasoning allows us to infer that location L_2 is contained in the buffer zone B and L_3 is contained in a storage zone S, whereas we already had the knowledge about L_1 belonging into the entrance zone E.

4 FORMALIZATION

The descriptions and formulas are based on prior work [19].

4.1 Formalizing the Warehouse Scenario

To introduce our notion of uncertain we need the formal basis for our process recognition. Processes and general background knowledge is modeled in LTL formulas (see Sec. 4.3) which capture the characteristics of spatio-temporal processes. These formulas are the robots observation interpretation into logic and have a twofold use: abstraction from possible noisy sensor measurements and knowledge transfer from metric data into a more human readable format. The grounding of our primitives is read as follows:

goods: A set $\mathcal{G} = \{G_1, \ldots, G_n\}$ of goods constitutes the entities that move in space over time and determine the dynamics of the scenario. They are observable by the robot and their position can be estimated.

locations: A location is a property of a good which remains the same when a good is not moved. During spatio-temporal grounding, position estimates are abstracted to a discrete set of locations. For a spatially restricted scenario the set of locations $\mathcal{L} = \{L_1, \ldots, L_m\}$ is finite.

zones: The warehouse scenario involves *functional zones* $\mathcal{Z} = \{E, B, S, P, O\}$ as described in Section 3.1. The extent of a zone is defined by the set of locations it contains. Zones are considered to be fix in our scenario, but their extent is a-priori unknown to the reasoning system.

4.2 Atomic Propositions for Spatio-Temporal Processes

Our atomic propositions comply with common syntax used for description in LTL. Construction of this atomic propositions is either done by said abstraction from sensor data or logic inference. We utilize the following atomic propositions which we denote in a predicate style for ease of readability, i.e., the atom $\mathrm{at}(G, L)$ stands for $|G| \cdot |L|$ atoms, one per combination of good G and location L.

- $\mathrm{at}(G, L) \Leftrightarrow$ good G is at a location L.
  This type of atom is data-driven, that is, its value can directly be obtained from sensor observations of the robot. Proposition $\mathrm{at}(G, L)$ holds if and only if a good G is known to be at location L. Truth of this proposition can thus change over time if a good is moved.
- $\mathrm{in}(L, Z) \Leftrightarrow$ location L is contained in a zone Z.
  As the set of locations is generated at runtime, $\mathrm{in}(L, Z)$ also depends on sensor perceptions. The interpretation of $\mathrm{in}(L, Z)$ remains constant over time.
- $\mathrm{close}(L_1, L_2) \Leftrightarrow$ two locations L_1, L_2 are close to one another.
  We use closeness as a central concept to distinguish different zones. $\mathrm{close}(L_1, L_2)$ remains constant over time.

Furthermore, the following constraints are in effect: A good can only be at one location at a given time. Locations do not move around, which implicates that one location that is close to another one stays close. A location that belongs to a zone does not change its zone. Zones itself are static, do not move and do not overlap.

For a in depth description of the integrity constraints associated with the atomic propositions, see [19].

4.3 Linear Temporal Logic (LTL)

Linear Temporal Logic [24] extends a propositional logic by temporal operators in a formula ϕ. The set of the usual logic operators includes $\wedge, \vee, \neg, \rightarrow$ and is enhanced by:

- $\circ\phi$ – next. A formula ϕ holds in in the following world
- $\Box\phi$ – always. A formula ϕ holds now and in all future worlds
- $\Diamond\phi$ – eventually. ϕ will hold in some world in the future ($\Diamond\phi \leftrightarrow \neg\Box\neg\phi$)

In our approach, we make extensive use of the $\Diamond$ operator to connect a spatial state transition of a good to a later point of time and therefore make use of LTL.

4.4 In-Warehouse Processes

Three process definitions are predominant in our warehouse scenario: *admission*, *take-out*, and *redistribution* of goods.

- *Admission* – a good G is delivered to the warehouse's entrance zone E and moved to the storage zone S via the buffer zone B. For all $G \in \mathcal{G}$ and $L_i, L_j, L_k \in \mathcal{L}$:

$$\begin{aligned}&\text{Admission}_{G,L_i,L_j,L_k} = \text{at}(G, L_i) \wedge \text{in}(L_i, E) \wedge \\ &\Diamond\Big(\text{at}(G, L_j) \wedge \text{in}(L_j, B) \wedge \Diamond\big(\text{at}(G, L_k) \wedge \text{in}(L_k, S)\big)\Big)\end{aligned} \tag{1}$$

- *Take-out* – a good G is moved from the storage zone S to the outlet zone O via a picking zone P. For all $G \in \mathcal{G}$ and $L_i, L_j, L_k \in \mathcal{L}$:

$$\begin{aligned}&\text{Takeout}_{G,L_i,L_j,L_k} = \text{at}(G, L_i) \wedge \text{in}(L_i, S) \wedge \\ &\Diamond\Big(\text{at}(G, L_j) \wedge \text{in}(L_j, P) \wedge \Diamond\big(\text{at}(G, L_k) \wedge \text{in}(L_k, O)\big)\Big)\end{aligned} \tag{2}$$

- *Redistribution* – a good G is moved within the storage zone S. For all $G \in \mathcal{G}$ and $L_i, L_j \in \mathcal{L}, i \neq j$:

$$\begin{aligned}&\text{Redistribution}_{G,L_i,L_j} = \text{at}(G, L_i) \wedge \text{in}(L_i, S) \wedge \\ &\Diamond\big(\text{at}(G, L_j) \wedge \text{in}(L_j, S)\big)\end{aligned} \tag{3}$$

4.5 Inferring Functional Zones

Our domain encloses functional zones, which are used to spatially differentiate states of our process objects. While there may be knowledge of zones beforehand, our deductive model checking approach works by assigning missing zones. So to speak, we are looking for models that satisfy our process definitions of affiliation to given zones at given points of time. Current practice in [19] is that for each good a history is searched for, independent of zones assigned while matching a history for a different good.

A new contribution in this paper is that our reasoning, while keeping the functionality of the model checking, is now able to take inferred zone knowledge from other goods, e.g. the mapping of free zones to functional zones, into account for all processes taking part in the environment.

5 HYPOTHESES GENERATION

Our knowledge base consists until now of facts which are abstracted from observations, for example, an occurrence of an entity at a definite location at a definite time. These facts are grounded on spatial and temporal context, which suggests that hypotheses have to be likewise. So far observed facts were transferred into at() atoms and used as entries into the model checker.

Taking the at() atomic part of the logic formula (1) from before we can define a method for generation of additional hypothetical facts. For the hypothetical facts, a notion of uncertainty to rate them is needed. As our formulas are logic based, a model for uncertainty that keeps the benefits of human understandability and abstraction is desired. That results in the following qualitative notion of decreasing certainty about a fact. The syntax is comparable to fuzzy logic concepts, albeit I do not have certainty intervals that do overlap in parts. The following formula C maps an atomic part ϕ of a logic process description to a uncertainty value:

$$\begin{aligned}&C(\phi) = \{true, certain, uncertain, \\ &doubtful, false\}\end{aligned} \tag{4}$$

Beginning from the left side, likelihood descriptions decreases stepwise.

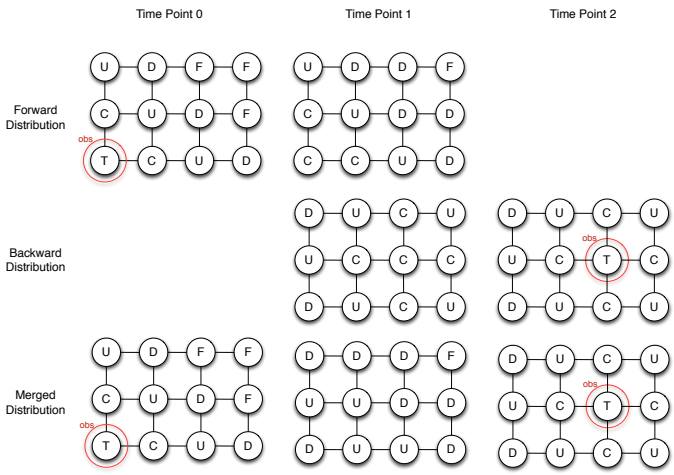

Figure 3. Example: Certainty propagation over 3 time steps. Upper line shows the forward derived step from time point 0 to 1, line 2 shows the backward derived step from time point 2 to 1 and line 3 shows the merged intermediate step between two observations. The variables (T, C, U, D, F) correspond to equation 4.

An instance of this mapping is done when actually observed facts are added to the logic knowledge base. An observed good G at the Location L_i is leveled $true$:

$$C(\text{at}(G, L_i)) = true \tag{5}$$

For different combinations of knowledge facts in the reasoning process appropriate mechanisms for combination can be written. The combination of two atomic facts ϕ, ψ result in the minimum certainty of the single facts for our first setup.

$$C(\phi \wedge \psi) = min(C(\phi), C(\psi)) \tag{6}$$

In this paper, derived hypothetical facts are rated less certain than observed facts. The approach is to relate bigger differences in space and time to a lower certainty.

5.1 Hypotheses by Spatial Deduction

Currently, three further hypotheses variations of the at() atomic fact are present. An observation at() of a fact close to (L_k) is still leveled $certain$ and an iteration one more close() relation away (L_m) is rated $uncertain$. If there is no observation of a good at a given time point at any location of the list of locations (Loc) a default hypotheses is generated for all locations with a rating of $doubtful$. Only one hypothesis per good, location and time point is allowed and the most certain option is kept.

$$C(\text{at}(G, L_i)) = \begin{cases} \textbf{if } \text{close}(L_i, L_k) \wedge \text{at}(G, L_k) & = certain \\ \textbf{if } \text{close}(L_i, L_k) \wedge \text{close}(L_k, L_m) & \\ \wedge\, \text{at}(G, L_m) & = uncertain \\ \textbf{if } (\neg\text{at}(G, L_j), L_j \in Loc) & = doubtful \end{cases}$$

An example of that uncertainty propagation is depicted in the upper left element of Figure 3, where you can see an observation at a location marked by a red circle and by the close() relation connected locations with derived uncertainties. The upper line depicts the forward propagation of uncertainty based on equation 5.1, the middle

line shows the backward propagation and the lower line is the merge from both time directions. For now, the method to combine formulas with uncertainty is to weight the result based on the lowest certainty used, e.g. a formula with one $certain$ and one $uncertain$ element will result in an $uncertain$ outcome. When more than one element of the lowest likelihood is present, this automatically results in the same likelihood, e.g. a formula with two $uncertain$ facts generates a $uncertain$ result. Facts are only added to the knowledge base, e.g. no evaluation of observations is made. In a future step a sanity check for observations is planned, that takes for example distance traveled in a given time as an argument and hands out a corresponding certainty if that movement is likely.

5.2 Hypotheses by Temporal Deduction

A similar approach is chosen for the temporal component, based on the timestamps, that denote an actual change in our knowledge base after entered into the reasoner, not observed goods are reentered with a reduced certainty. Certainty is decreased until is reaches the default level over time steps (T).

$$C(\text{at}(G, L_i)) = \begin{cases} \textbf{if } \neg\text{at}(G,L)_T \wedge \text{at}(G,L)_{T+1} & = certain \\ \textbf{if } \neg\text{at}(G,L)_T \wedge \text{at}(G,L)_{T+2} & = uncertain \\ \textbf{if } \neg\text{at}(G,L)_T \wedge \text{at}(G,L)_{T>2} & = doubtful \end{cases}$$

5.3 Uncertainty for Combined Facts

A further step that integrates uncertainty is the combination of facts. Currently there are two combinations that take uncertainty into account. Firstly the combination of facts that combine into a process ($admission$, $redistribution$ or $takeout$) and the allocation of an unbound zone to a functional zone. Both orientate on the default in equation 6 and hand back the lowest uncertainty used in its atoms.

6 EVALUATION

The data used in this evaluation is in form of logic atoms that are a simplified version from the abstracted facts taken from recordings of the same warehouse environment as used in our previous paper. Due to a straight forward Prolog implementation[2], complexity is still an issue. Nevertheless, the following results are (shorted) output from the Prolog code. This evaluation shows the feasibility of my approach and displays a list of outcomes that lead to new open questions. Our scenario consist of at() atoms, close() atoms, a list of locations $locs$ and a list of wares $wares$. Previous knowledge about functionality of zones is given as entrance and outlet are known (this is a given in most real warehouses), locations not close to each other can belong to different zones.

Inference on a completely observed process

observations	old algorithm	new algorithm
at($good_1, loc_1$)	found Histories:	found Histories:
at($good_1, loc_2$)	[$good_1$, [[$entrance$,	[$good_1$, [[$entrance$,
at($good_1, loc_3$)	$buffer, storage$,	$buffer, storage$,
at($good_1, loc_4$)	$picking, outlet$]]	$picking, outlet$],
at($good_1, loc_5$)		$true$]]

This table shows a result for a run with a single good and 5 different observed locations for the algorithm in [19] and the algorithm from this paper. Both runs detect the full history ($admission, takeout$, composed of one stop at each of the 5 functional zones) as every stop of the good in the warehouse is observed. Note the missing certainty information of the old result, based on the fact that no uncertainty estimation was made.

[2] http://www.swi-prolog.org/, my source code is available on request.

Inference on a partially observed process with one missing observation

observations	old result	new result
at($good_1, loc_1$)	found Histories:	found Histories:
	[$good_1$, [[$storage$	[$good_1$, [[$entrance$,
at($good_1, loc_3$)	$picking, outlet$]]	$buffer, storage$,
at($good_1, loc_4$)		$picking, outlet$],
at($good_1, loc_5$)		$doubtful$]]

This table shows a result for a run with a single good and 4 different observed locations. Note that the old approach did only recognize a $takeout$ (see equation 2) from the combination of zone knowledge and inference, while the new algorithm makes use of a substitution for the missing location and is able to infer the whole history. As the uncertainty for the substitution is rated $doubtful$, the whole process uncertainty drops to that level.

Inference on a partially observed process with one misplaced observation

observations	old result	new result
at($good_1, loc_1$)	found Histories:	found Histories:
at($good_1, loc_6$)	[$good_1$, [[$storage$	[$good_1$, [[$entrance$,
at($good_1, loc_3$)	$picking, outlet$]]	$buffer, storage$,
at($good_1, loc_4$)		$picking, outlet$],
at($good_1, loc_5$)		$certain$]]
closeTo(loc_2, loc_6)		

Here, additional information in form of a $closeTo$ relation is given. A substitution for the missing at relation is formed and the process uncertainty raises to $certain$, conforming to algorithm 5.1.

Inference on a partially observed process with two missing observations

observations	old result	new result
at($good_1, loc_1$)	found Histories:	found Histories:
		[$good_1$, [[$entrance$,
at($good_1, loc_3$)		$buffer, storage$,
		$picking, outlet$],
at($good_1, loc_5$)		$doubtful$]]

This table depicts a situation, where a process is only observed in the entrance, the storage itself and the takeout area. Our original approach is not able to match a single process description as critical observations are missing. The hypotheses enhanced approach can fill the missing pieces, but results in a $doubtful$ uncertainty.

7 DISCUSSION & OPEN QUESTIONS

Our prototypical Prolog implementation shows that the tradeoff of definitely detected processes against detected processes with a uncertainty value yields additional benefits. Processes, already detectable by the old algorithms, are detected true and all remaining detected processes have not been seen by the old approach. The price we pay for an enhanced detection rate is computational complexity. Cautious

estimations for the added complexity have to consider a worst case scenario, where $doubtful$ hypotheses are inserted at every known location, as long as no observation is made for a ware.

This problem can be reduced by a more sensible construction of hypotheses, which is still at a basic level. One open research question here is to improve hypotheses generation, especially by abductive based methods. Another possible solution is found in use of heuristics for the model checking. It is to be investigated if we need every solution or just an optimal one. In the experiments, we observe that not in every case the integration of a $doubtful$ fact, that reduces the overall process uncertainty to doubtful actually makes sense. When comparing the integration of one or more facts with the same level of uncertainty, differences should be made accordingly to the quantity of taken hypotheses. That opens up another research question of a better integration of hypotheses in the deductive model checking. Yet another open issue is the verification of observations. Integration of an evaluation how likely an actual observation is contributes to robustness against sensor noise and may drop the number of facts that the model checker has to utilize.

8 CONTRIBUTION

Our contribution so far is twofold, we propose a way of hypotheses generation to close gaps in surveillance data used for process recognition and realized the inferenz reasoning for process recognition that takes deducted zone knowledge for found histories of goods into account. While the integration of inference knowledge into the reasoning process will lead to better overall results as more false positives vanish due to the cross validation of our processes, far more potential lies in the use of hypotheses for the reasoning process. By adding more validation mechanisms to the generation procedure to get more reliable hypotheses, the number of hypotheses will decrease and complexity will drop. But even without the upcoming enhancements, the introduction of hypotheses into the reasoning process has resulted in a more accurate detection rate.

ACKNOWLEDGEMENTS

This paper presents work done in the project R3-[Q-Shape] of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition. Financial support by the German Research Foundation (DFG) is gratefully acknowledged.

REFERENCES

[1] F. Abrate, B. Bona, M. Indri, and L. Carlone, 'Cooperative robotic teams for supervision and management of large logistic spaces: Methodology and applications', in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pp. 1–8, (2010).

[2] Maria-Florina Balcan and Avrim Blum, 'A discriminative model for semi-supervised learning', *Journal of the ACM (JACM)*, **57**(3), 1–46, (2010).

[3] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun, 'Learning motion patterns of people for compliant robot motion', *The International Journal of Robotics Research (IJRR)*, **24**(1), 39–41, (2005).

[4] Bert Bredeweg and Peter Struss, 'Current topics in qualitative reasoning', *AI Magazine*, **24**(4), (January 2004).

[5] H.H. Bui, 'A general model for online probabilistic plan recognition', *International Joint Conference on Artificial Intelligence*, (2003).

[6] Alessandro Cimatti, Fausto Giunchiglia, Enrico Giunchiglia, and Paolo Traverso, 'Planning via model checking: A decision procedure for *AR*', in *European Conference on Planning (ECP)*, eds., Sam Steel and Rachid Alami, volume 1348 of *Lecture Notes in Computer Science*, pp. 130–142. Springer, (1997).

[7] Krishna Dubba, Mehul Bhatt, Frank Dylla, Anthony Cohn, and David Hogg, 'Interleaved inductive-abductive reasoning for learning event-based activity models', in *Inductive Logic Programming - 21st International Conference, ILP 2011. Windsor Great Park, UK.*, Lecture Notes in Computer Science, (2011).

[8] Didier Dubois and Henri Prade, 'Possibility theory, probability theory and multiple-valued logics: a clarification', *Annals of Mathematics and Artificial Intelligence*, **32**(1-4), 35–66, (2001).

[9] Stefan Edelkamp and Shahid Jabbar, 'Action planning for directed model checking of petri nets', *Electronic Notes in Theoretical Computer Science*, **149**(2), 3–18, (February 2006).

[10] R Fagin, J.Y Halpern, and N Megiddo, 'A logic for reasoning about probabilities', in *Logic in Computer Science, 1988. LICS '88., Proceedings of the Third Annual Symposium on*, pp. 410–421, (1988).

[11] Forbus, 'Qualitative reasoning', *The Computer Science and Engineering Handbook*, (1996).

[12] M Goldszmidt, 'Qualitative probabilities for default reasoning, belief revision, and causal modeling', *Artificial Intelligence*, (1996).

[13] Torsten Hildebrandt, Lutz Frommberger, Diedrich Wolter, Christian Zabel, Christian Freksa, and Bernd Scholz-Reiter, 'Towards Optimization of Manufacturing Systems using Autonomous Robotic Observers', *Proceedings of the 7th CIRP International Conference on Intelligent Computation in Manufacturing Engineering (ICME 2010)*, (2010).

[14] A Ilić-Stepić, 'A logic for reasoning about qualitative probability', *Publications de l'Institut Mathematique*, (2010).

[15] Marius Kloetzer and Calin Belta, 'LTL planning for groups of robots', in *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pp. 578–583, (2006).

[16] Marius Kloetzer and Calin Belta, 'Automatic deployment of distributed teams of robots from temporal logic motion specifications', *IEEE Transactions on Robotics*, **26**(1), 48–61, (2010).

[17] C Kowalski, 'Using adaptive neural networks for situation recognition in high and low-intensity conflict', in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, (1991).

[18] Hadas Kress-Gazit, Tichakorn Wongpiromsarn, and Ufuk Topcu, 'Correct, reactive robot control from abstraction and temporal logic specifications', *Special Issue of the IEEE Robotics and Automation Magazine on Formal Methods for Robotics and Automation*, **18**(3), 65–74, (September 2011).

[19] A. Kreutzmann, I. Colonius, L. Frommberger, F. Dylla, C. Freksa, and D. Wolter, 'On Process Recognition by Logical Inference', in *European Conference on Mobile Robots*, pp. 1–6. SFB/TR 8 Spatial Cognition, University of Bremen, Germany, (July 2011).

[20] Morteza Lahijanian, Sean B. Andersson, and Calin Belta, 'Temporal logic motion planning and control with probabilistic satisfaction guarantees', *IEEE Transactions on Robotics*, **PP**(99), 1 –14, (2011).

[21] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz, 'Learning and inferring transportation routines', *Artificial Intelligence*, **171**(5–6), 311–331, (2007).

[22] T Magherini, G Parente, C.D Nugent, M.P Donnelly, E Vicario, F Cruciani, and C Paggetti, 'Temporal Logic Bounded Model-Checking for recognition of activities of daily living', in *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on*, pp. 1–4, (December 2010).

[23] Fulvio Mastrogiovanni, Antonio Sgorbissa, and Renato Zaccaria, 'Context assessment strategies for ubiquitous robots', in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2717–2722, (2009).

[24] Amir Pnueli, 'The temporal logic of programs', in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 46–57, (1977).

[25] Stephen L. Smith, Jana Tůmová, Calin Belta, and Daniela Rus, 'Optimal path planning under temporal logic constraints', in *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3288–3293, Taipeh, Taiwan, (October 2010).

[26] Michael Ten Hompel and Thorsten Schmidt, *Management of Warehouse Systems*, chapter 2, 13–63, Springer, Berlin Heidelberg, 2010.

[27] Qiang Yang, 'Activity recognition: Linking low-level sensors to high level intelligence', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 20–25, Pasadena, CA, (July 2009).