Journal of Universal Computer Science, vol. 8, no. 8 (2002), 808-832
submitted: 11/2/02, accepted: 29/4/02, appeared: 28/8/02 © J.UCS

Scheduling Tasks to a Team of Autonomous Mobile Service Robots in Indoor Enviroments

Hartmut Surmann
(Fraunhofer Institute for Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany,
surmann@ais.fhg.de)
Antonio Morales
(Jaume-I University, Robotic Intelligence Laboratory
Campus Riu Sec, Edificio TI, E-12071 Castellon, Spain,
morales@inf.uji.es)

Abstract: This paper presents a complete system for scheduling transportation orders to a fleet of autonomous mobile robots in service environments. It consists of the autonomous mobile robots, a user friendly interface to acquire the orders for the robots via internet and to store them in a database, a general language for modeling multistorey buildings with XML and the scheduling algorithms. The model description of the buildings is used to plan the paths for the robots and to estimate the cost and times for the orders. One challenging key problem – the multi robot cooperation – is solved by the scheduling algorithms and by giving *autonomy* to the service robots.
Key Words: Job scheduling, autonomous mobile service robots, robot teams human machine interface (HMI), NP-hard problems, XML, telerobotics.
Category: I.2.9, D.4.1

1 Introduction

Interest in autonomous mobile service robots continuously increases since J. Engelberger present his transport service robot HelpMate [Eng93, SV96]. Service robots have to achieve a high level of flexibility, adaptability, and efficiency in human-populated areas. Today, only few service robot systems are available. Some are built for transportation and delivery tasks like serving drinks in a hotel [RG98], delivering mail [Ves96], or offering transportation capacity in a hospital [Eng93].

In contrast to these single robots approaches, this paper presents a team of currently three service robots. They offer the possibility to simultaneously handle a large number of tasks in multistorey buildings in a predefined time window or interval. Therefore, a large number of transportation tasks have to be acquired, organized and scheduled to the service robots.

For transportation tasks, this problem is known as the vehicle routing problem (VRP). It involves the design of a set of minimum cost routes for a fleet of vehicles, e.g., service robots. The VRP area has been intensively studied in literature. We refer the reader to [LBB83, LN87, Mag81] for comprehensive surveys on the VRP and its variations that also describe many practical occurrences of the problem.

In the VRP with time windows (VRPTW) the above issues have to be dealt with under the added complexity of specified delivery times or time windows. The pickup and delivery problem with time windows (PDPTW) is a generalization of the VRPTW. It is concerned with the construction of optimal routes to satisfy transportation requests, requiring both pickup at the origin and delivery at the destination under capacity, time window and precedence constraints. In addition, each route satisfies pairing constraints since corresponding pickup and delivery locations have to be served by the same vehicle. The VRPTW is a particular case of the PDPTW where the destinations is always one common depot. Solomon et al. give a recent surveys on time window constrained routing and scheduling problems [SD88, YDS91, SW00].

An important constraint for the PDPTW is the consideration of the vehicle capacity. In the preemptive case, the vehicles have an infinite capacity and could be interrupted to pickup new items. In the non-preemptive case, the route from the pickup location to the corresponding delivery location could not be interrupted. The ARIADNE robots have a limited transportation capacity and can pickup from 1 two 20 objects depending on the object size. Therefore, a scheduling algorithm is needed which considers all possibilities between the preemptive and the non-preemptive case.

In contrast to industrial environments, a further important constraint for a transport application in service environments is the customer orientation. The vehicle serves a set of customers with permanently varying demands. Customer orientation is more than to fulfill the transportation request in time. Many additional items specify the customer needs.

- A simple and comfortable user interface. Transport tasks can be initiated at any time and at any computer.
- An always up-to-date database for the orders and schedules with an Internet connection to ensure transparency.
- The up-to-date constraint leads to soft real-time behavior of the scheduling algorithm, e.g., it has to work online.
- If possible, previous schedules have to be changed only moderately to achieve planning certainty for the customer. This condition will be called **minor changing constraint**.
- Shortest possible routes to gain high throughput, low energy consumption and a short response time.
- In addition to the route planning a time estimation is also demanded.

– Privacy, i.e., orders should be seen only by the originator.

Since practical applications in industrial environments are static, i.e., the number of vehicles and the pickup and delivery locations are fixed, known in advance and constant over the time, previous work on PDPTW concentrated mainly on the minimization of the length of the route [SD88]. For this type of application, the calculation time of the scheduling algorithm is secondary.

Other similar projects are the Martha project [RAHIR98]. The objectives are the operation and control of a large fleet of autonomous mobile robots (10-100) for containers transshipment tasks in harbors, airports and marshaling yards. Simmons et al. use the Prodigy task scheduler for the Xavier robot [SGH$^+$97]. Prodigy is a domain-independent nonlinear problem solver that uses means-ends analysis and backward chaining to reason about multiple goals and multiple alternatives of achieving them. Rogue, an architecture that integrates high-level planning with a low-level executing, is the framework that integrates the Xavier robot with the prodigy planning system [HV97]. Burghard and Thrun et. al. build a scheduling/planning system with a web interface for their solitary acting museum robots MINERVA and RHINO [BCF$^+$98, TBB$^+$99]. Even though all of these robots (including ours) work autonomously, the approaches could be regarded as tele-robotics, tele-controlled or better tele-instructed with a high degree of autonomy for each robot.

In this paper we present the complete system for scheduling transportation orders to autonomous mobile robots in service environments. Therefore, we introduce the mobile service robots in section 2. Section 3 describes the customer interface to receive transportation orders and section 4 gives an example of a prototype multistorey building and introduces the XML language for modeling buildings. Section 5 presents the online capable algorithms for the preemptive and non-preemptive scheduling strategies. A conclusion forms the last chapter of the paper.

2 Service robot team - ARIADNE

The ARIADNE team consists of three mobile industrial robots (Fig. 1): Odysseus, Marvin and Thor. Each is about 80 cm $\times$ 60 cm large and 90 cm high. The mobile platform can carry a payload of 200 kg at speeds of up to 0.8 m/s (about half the speed of a pedestrian). The right and left driving wheels are mounted on a suspension on the center line of the mobile platform. Passive castors on each corner of the chassis ensure stability.

The core of every robot is a Pentium PC 166 MHz with 16 MB RAM and real-time Linux. Two micro-controllers are used. One controls the internal states, display, keyboard and radio link of the robot. The other one manages the motors and the optical line-tracking. Each platform is rigged with two laser scanners,

Figure 1: Left: The ARIADNE-team: Odysseus, Marvin, and Thor executing service orders. Each robot is equipped with four drawers, two on each side. Right: A robot automatically charges its batteries.

one on the front and the other on the rear of the robot. Each laser scans 180° of the environment [PSFL98].

Each of the 250 kg robots can operate for about 8 hours with one battery charge. When the power drains the robot visits an automatic power recharging station, connects itself to it and recharges its batteries (Fig. 1, right). A behavior based approach is used to control the robots [SP01]. Behaviors are expressed by groups of fuzzy rules and different behaviors are combined by the compositional rule of inference [TW86, SHP95, Saf97]. Important behaviors are obstacle avoidance, wall following, robots meets human and robot meets robot. Planning advises are combined with the behaviors by using fuzzy states in the fuzzy rules [SHW96]. Refining abstract goals, e.g., going to an office is done by the robots. Current information about the robot, e.g., robot monitoring, robot orders, schedules, battery loading, laser scans etc., can be monitored at http://lamu.gmd.de:8080.

Each robot sends a heartbeat, i.e., a position change information while it is moving and some other signal if it is standing, e.g., while charging its battery. If the heartbeat stops and the robot is unreachable, the administrator will be alerted immediately. If the unavailable robot was executing some task, the administrator can schedule this task to another robot. An additional task might be necessary for finding the defective robot and unloading the robot cargo. The behavior of the robots is visualized two and three dimensional with a Java applet or a VRML browser. The applet and the VRML browser increase the user friendliness of the system (Fig. 2). Additional live cameras assist the administrator to monitor the robots and to detect critical situations, e.g., an area crowded by persons or an area of construction.

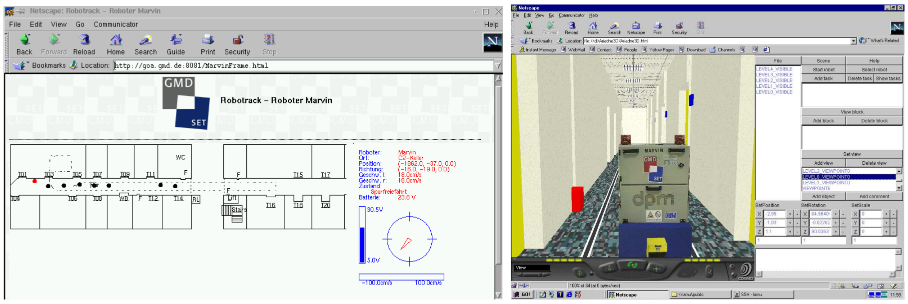


Figure 2: 2D and 3D monitoring window.

The key idea behind the development of the team is to give more *autonomy* to the service robots and allow them to cope with unexpected events and obstacles, inaccurate environment model, other vehicles, and so on. This high level of autonomy is achieved using advanced sensor-based capabilities, e.g., for localization, obstacle detection and modeling, as well as planning and deliberation between the robots through local communication and coordination [SM00].

3 Customer interface

The customer interface (RoboDis) is a client-server web-based dispatching system for multiple autonomous service robots executing transport orders. The web interface allow users to enter transport orders which consist of a pickup and a delivery event and to store them in a SQL database. The scheduling algorithms load the list of orders from the database and schedule the orders. The scheduled events are stored in a second database where each robot has its own event list. If a robot has finished an event or at regular time intervals a robot contacts the second database client to get a new event (pickup or delivery) via its radio link connection.

Transport orders for the service robots can be entered by means of a standard WWW client such as Netscape Navigator or Internet Explorer. In our system more pretentious requirements are made for the term service, i.e., the total system must be highly available, user friendly, safe and transparent. Transparency means, a simple view of the system is given to the user: a specific list of transport orders is defined by the user and executed in time. System complexity, e.g., specifying which robot and how a robot executes an order, is hidden from the user. [1]

[1] The dispatching system can be tested under URL http://lamu.gmd.de:8080/. Unregistered users may enter transport orders using the account guest with password guest. However these orders are not executed by the robots.

In order to implement such a universal dispatching system the reliability of the system is important. Among reliability the following service and safety aspects are summarized:

– High availability of the system: The system should be at disposal 24 hours a day to all users.

– Authentication of the users: The user who charges the system with a transport order, must be uniquely identifiable. 180 computers in our offices are connected to the system.

– Privacy: Encryption of the transport orders. Users may access only the orders charged by themselves.

– Integrity of the transport orders: The web server has to receive the orders exactly as they were sent by the user.

– System administration: a large system with numerous robots must support treatment of unforeseen situations and overcoming system errors.

Figure 3 illustrates physical and software components of the system. The mission server fetches unfinished orders and schedules them for execution by some robot depending on the availability, the position of the robot and on the execution time window of the order defined by the user. The radio server connects the other system components through radio links, e.g., it transmits orders to the robots, informs the mission server about robot positions and notifies the door and elevator server about the state of each door and elevator respectively. The door server stores the state of each door and decides when to change it, i.e., when to open/close doors. The elevator server similarly operates for the elevators, e.g., when a robot requests an elevator, the server decides whether this is in conflict with human transportation requirements of the elevator and eventually serves the robot.

While each robot forms a control unit of its own, other control units like the elevator server support more than one physical component. This reduces complexity by reducing the number of system components. In a system with a larger number of physical components, several server instances can run parallel, e.g., one elevator server for each building. A cluster structure can be used for the parallel servers to keep communication requirements low.

3.1 High availability

A user enters transport orders through a web interface (Fig. 4). Among other things, redundancy provides high availability. Several PCs can be used as a hardware platform for the web server. Here, standard PCs with the Linux operating system (v. 2.4.4) and Apache web server (v. 1.3.19) have been selected.

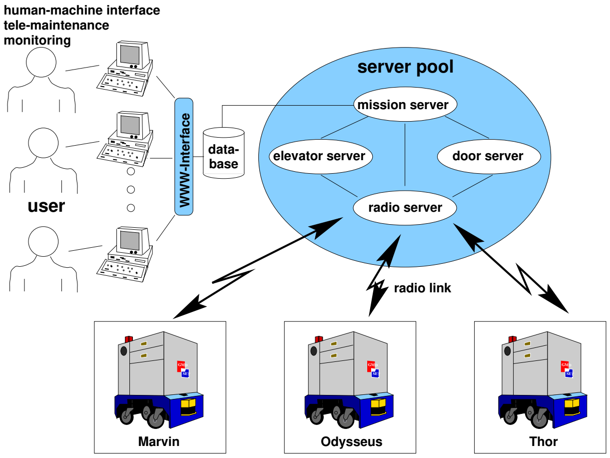


Figure 3: System structure overview

Figure 4: Web formula for transport orders

Linux provides a fast and reliable disk system by supporting software RAID [Vep99]. RAID level 5 allows the system to continue an operation even if one of its hard disk fails. The database ensures the integrity of the transport orders with its atomic transactions: each order is either stored completely or not at all. Even in case of a failure there is no in-between state with corrupted data.

The Apache server as well as the stateless HTTP-protocol increase the availability because inquiries are forked to independent child processes. If one of these child processes dies, the total system is not endangered. A further aspect for high availability is the information throughput, which amounts up to 1000 inquiries per minute for the current implementation. The Apache server is world-wide the most frequently used web server. In particular the source code is available and therefore its bugs are well-known and documented.

Software components of the building management system are implemented as CGI scripts [DG96], [Inc98]. This additionally increases the service. So on the one hand, data (orders, robot positions, states of doors and elevators etc.) can be stored in and queried from a database. Detailed debugging protocols can be created from the database if necessary. On the other hand a modular system is achieved, which can be extended de-centrally. Further components are easily added by implementing new CGI scripts (add-ons).

3.2 Privacy

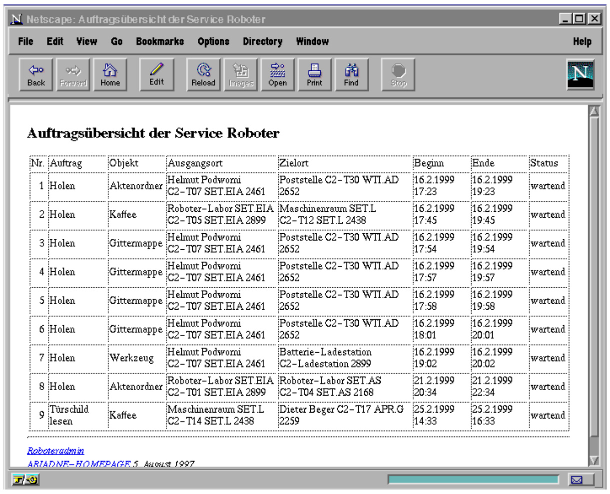
Privacy is achieved by encryption. Since a web-based client-server system is realized, the Secure Socket Layer protocol (SSL) [Cor98],[Lau98] is used for the encryption. All data that requires an Authentication is transmitted encrypted by the client to the server. So transport orders, order lists (Fig. 5), deletions of orders, passwords, password modifications etc. are protected against reading by unauthorized persons. The SSL protocol is particularly user friendly, since the encryption is done by the browser and web server and is transparent for the user. The software in use is OpenSSL 0.9.6b with the appropriate Apache patch. The server uses self-created certificates. On server side encryption up to 128 bits is possible, if the browser supports this. Otherwise the keys have a minimum length of 40 bits.

4 The experimental environment

The robots currently move in the GMD-ROBOBENCH, an typical multistorey H-shaped building of about 1600 m^2 (Fig. 6) which servers as a general prototype for other multistorey buildings. The building is departed into 84 office rooms and 15 corridors on 3 levels reachable by elevators. All elevators and 17 fire protection doors can be automatically controlled by every robot via an internet radio link. Office room doors can be opened by humans only. Web cameras on some floors allow live observation of robot actions.

4.1 A language for modeling the buildings

The spatial geometry and topology of the buildings is the base for the scheduling algorithms and has to be modeled. Modeling is the name of the game in any intelligence, be it human or machine. With the model and its exercising we can



Netscape: Auftragsübersicht der Service Roboter

Auftragsübersicht der Service Roboter

Nr.	Auftrag	Objekt	Ausgangsort	Zielort	Beginn	Ende	Status
1	Holen	Aktenordner	Helmut Podwomi C2-T07 SET.EIA 2461	Poststelle C2-T30 WTI.AD 2652	16.2.1999 17:23	16.2.1999 19:23	wartend
2	Holen	Kaffee	Roboter-Labor SET.EIA C2-T05 SET.EIA 2899	Maschinenraum SET.L C2-T12 SET.L 2438	16.2.1999 17:45	16.2.1999 19:45	wartend
3	Holen	Gittermappe	Helmut Podwomi C2-T07 SET.EIA 2461	Poststelle C2-T30 WTI.AD 2652	16.2.1999 17:54	16.2.1999 19:54	wartend
4	Holen	Gittermappe	Helmut Podwomi C2-T07 SET.EIA 2461	Poststelle C2-T30 WTI.AD 2652	16.2.1999 17:57	16.2.1999 19:57	wartend
5	Holen	Gittermappe	Helmut Podwomi C2-T07 SET.EIA 2461	Poststelle C2-T30 WTI.AD 2652	16.2.1999 17:58	16.2.1999 19:58	wartend
6	Holen	Gittermappe	Helmut Podwomi C2-T07 SET.EIA 2461	Poststelle C2-T30 WTI.AD 2652	16.2.1999 18:01	16.2.1999 20:01	wartend
7	Holen	Werkzeug	Helmut Podwomi C2-T07 SET.EIA 2461	Batterie-Ladestation C2-Ladestation 2899	16.2.1999 19:02	16.2.1999 20:02	wartend
8	Holen	Aktenordner	Roboter-Labor SET.EIA C2-T01 SET.EIA 2899	Roboter-Labor SET.AS C2-T04 SET.AS 2168	21.2.1999 20:34	21.2.1999 22:34	wartend
9	Türschild lesen	Kaffee	Maschinenraum SET.L C2-T14 SET.L 2438	Dieter Beger C2-T17 APR.G 2259	25.2.1999 14:33	25.2.1999 16:33	wartend

Roboteradmin
ARIADNE-HOMEPAGE 5. August 1997

Figure 5: Order list

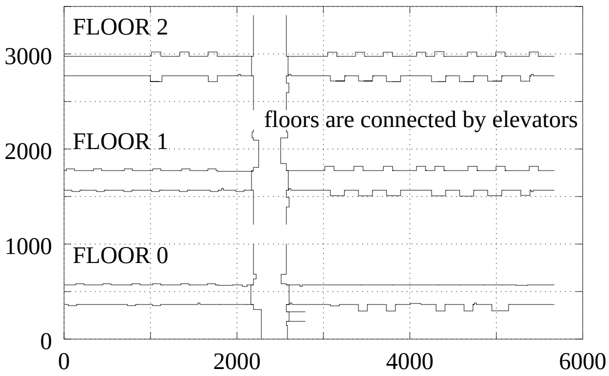


Figure 6: Layout of the prototype environment.

look forward in time with predictions and prescriptions and backward in time which diagnostics and explanations. With these time binding information structures we can make decisions and estimations in the here and now for purposes of efficiency, efficacy and control into the future. We and our machines hope to look into the future and the past so we may act efficiently now. Arkin pointed out that: *Deliberative systems permit representational knowledge to be used for planning purposes in advance of execution* and that *a significant controversy exists regarding the appropriate role of knowledge within robotic systems* [Ark98]. Our additional aim is to show the knowledge and data which has to be selected and how it can be organized to serve all modules from navigation over planning/scheduling up to the human machine interface (HMI) in time. In the case of scheduling orders for mobile robots in indoor environments it means that we have to model geometrical data, topological data and public data, e.g., names, telephone numbers, email addresses ... of the people who work in the building.

The model introduced here contains a topological map to schedule orders for robots as well as to plan robot behaviors. The planner is a graph planner [KS96]. In addition a geometrical and feature map for the landmark recognition and robot localization is implemented. The codes necessary for the handling of the elevators and doors are likewise stored in the model. Additional references to SQL databases are also included. The SQL database contains lists of persons with their surname, first name, email address, telephone number and door designator. Thus all information needed for the interaction of robots and humans is available and it is possible to implement a user friendly interface [ST99].

The important point is that the model is a general prototype for multistorey buildings and that it can be simple adapted to other buildings. Furthermore it can be extended easily to deal with dynamic objects. Figure 7 shows a segment of our language for the spatial modeling for multistorey buildings. Different sections are parenthetical by <name> ... </name> like the XML specification. Therefore, robots and/or other client/servers can exchange and update there spatial model over the Internet.

The "corridor"structure is a part of the topological map and used by the route planner. Each corridor has an internal unambiguous number, a name (C2-basement), a corridor type (0: office corridor, 1: junction corridor, 2: elevator ...), dimensions (width, length, height, orientation), a global position (x,y,z) and a list of connections to other corridors. Moreover, some control information for automatic doors is included, i.e.: "doorControlCmdOpen".

Further sections, e.g., <rightWall> or <leftWall> describes the geometric dimensions of the right resp. left wall of a corridor. A wall includes different objects, e.g., fire extinguisher, doors, door plates etc. It also can have protrusions or niches. The content of a door plate is important because it builds the reference to the SQL user database which includes telephone lists, email addresses etc. Additional parts, e.g., <Office> . . . </Office> describe the dimensions of offices and objects locate in the offices.

The geometric part of the world model could either be automatically generated from a floor plan or by a ride of the robot in the corridor. The names, corridor interconnection as well as the control codes for the elevator and fire doors have to added by hand.

One major point is that different internal and external descriptions of the world model are generated on the base of the spatial geometry, e.g., VRML for 3D robot tracking, 2D floor plans for gnuplot and a Java applet. The floor plans

```
<world name="GMD Building C2">
  <corridor name="C2-basement" color ="grey" type="0" lenght="2190" width="206"
                                                 height="300" orientation="-90">
    </globalReferencePoint posX="0" poxY="0" posZ="0">
    <connection name ="fire door" number = "4" color="indoor-4.jpg" posX ="-103"
                                                              posY ="0" posZ="0">
      <toCorridor>C2-crossing-corridor </entryPoint name="b"></toCorridor>
      <doorControlCmdOpen>8 82 4 1</doorControlCmdOpen>
      <doorControlCmdClose>8 82 4 0</doorControlCmdClose>
      <doorControlCmdStatus>8 82 4 2</doorControlCmdStatus>
    </connection>
    <connection name = "Outdoor" color="out-door.jpg" posX="-103" posY="2190"
                                                                       posZ="0">
      <toCorridor> empty </toCorridor>
    </connection>

    <rightWall name="" type="2" color="blue" lenght="2190" width="30"
                                                                   height="300">
      <DoorFrame name="" color="white" height="300" width="180" depth="-50"
                                                   posX="0" posY="200" posZ="0">
        <door name="" color="C2-116.jpg" height="300" width="80" depth="-50"
                                                   posX="0" posY="220" posZ="0">
          <doorPlate name="C2-116" color="C2-116p.jpg" posX ="-103" posY ="2190"
                                                                       posZ="0">
          <doorPlateContent>Hartmut Surmann, Andreas Nuechter, Kai Lingemann
            </doorPlateContent>
          </doorPlate>
        </door>
      <DoorFrame>
      <object name="fire extinguisher" color="fire-distinguisher.jpg">
        <cylinder radiusX = 20 height=40" posX="0" posY="420" posZ="30">
        <!--
          <sphere radiusX = "20" radiusY=40" posX="0" posY="420" posZ="30">
          <box lenght="40" height="30" width="30" posX="0" posY="420" posZ="30">
                                                                            -->
      </object>
      </protrusion name="" color="white" height="300" width="180" depth="-50"
                                                   posX="0" posY="870" posZ="0">
      ............
    </rightWall>
    <leftWall>
     .....
    </leftWall>
    .............
  </corridor>
</world>
```

Figure 7: Segment of the spatial model.

are also used for 2D tracking (Fig. 2) and simulations which is important for the design of the behavior modules as well as the prediction of the near future.

The plan of the environment will be explain by an example. Let as assume we have three corridors: Two office corridors, one connection corridor with an elevator (Fig . 8). The elevator is regarded as a special transport corridor, so at least we have to model four corridors. The two office corridors should have a length of 21.9 m, a width of 2.1 m and a height of 3 m. Both have one connection two the third corridor with the elevator. The connection corridor number 3 has two connections to the office corridor and one connection to the elevator. The elevator has a connection at each of the 3 layers.

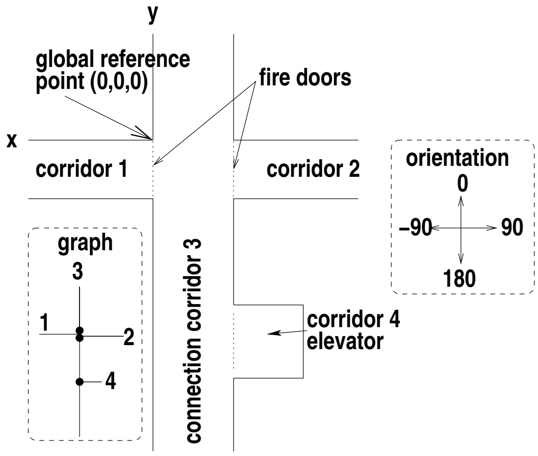


Figure 8: Example of the graph structure of the environment.

The following xml description describes the topological map:

<corridor name=”1” type=”0” lenght=”2190” width=”206” height=”300” orientation=”-90”>
</globalReferencePoint posX=”0” poxY=”0” posZ=”0”>
<connection name =”fire door” number = ”4” posX =”0” posY =”-103” posZ=”0”>
<toCorridor>3</entryPoint name=”a”></toCorridor>

<corridor name=”2” type=”0” lenght=”2190” width=”206” height=”300” orientation=”90”>
</globalReferencePoint posX=”380” poxY=”-206” posZ=”0”>
<connection name =”fire door” number = ”5” posX =”0” posY =”103” posZ=”0”>
<toCorridor>3</entryPoint name=”b”></toCorridor>

<corridor name=”3” type=”1” lenght=”380” width=”1000” height=”300” orientation=”0”>
</globalReferencePoint posX=”380” poxY=”-569” posZ=”0”>
<connection name =”fire door” number = ”4” posX =”-380” posY =”466” posZ=”0”>
<toCorridor>1</entryPoint name=”a”></toCorridor>
<connection name =”fire door” number = ”5” posX =”0” posY =”466” posZ=”0”>
<toCorridor>2</entryPoint name=”a”></toCorridor>
<connection name =”elevator” number = ”1” posX =”0” posY =”236” posZ=”0”>
<toCorridor>4</entryPoint name=”a”></toCorridor>

<corridor name=”4” type=”2” lenght=”200” width=”100” height=”900” orientation=”90”>
</globalReferencePoint posX=”380” poxY=”-283” posZ=”0”>
<connection name =”base” number = ”1” posX =”0” posY =”50” posZ=”0”>
<toCorridor>2</entryPoint name=”c”></toCorridor>

<connection name ="1 floor" number = "2" posX ="0" posY ="50" posZ="300">
<toCorridor>6</entryPoint ame="c"></toCorridor>
<connection name ="2 floor" number = "3" posX ="0" posY ="50" posZ="600">
<toCorridor>9</entryPoint name="c"></toCorridor>

The "Connection" section expresses the spatial relation between the different corridors and can be expressed in a graph like structure (Fig . 8). The "connection" definition is the basic description for the graph planner. The length, e.g., 21.9 m of the two office corridors is the base for the calculation of the costs of a schedule. Combined with the measured average speed of a robot, a time estimation can be done.

5 Scheduling algorithms

The term "planning" for mobile robots has been rightly subdivided into two questions: "Where am I going?" and "How do I get there?" [BF96]. Local and global path planning deal with these two questions. We extend these two questions by *"Where am I going first?"* and *"Why should I go there and not another robot?"*. The scheduling of multiple orders for a group of mobile service robots regards these questions and extends local and global path planning by a third block called scheduler.

The selection of possible scheduling algorithms is mainly restricted by two constraints: the soft real-time and the moderate change of previous schedules constraint. Especially the second constraint can not be guaranteed and conflicts with the minimization of the routes. These two contradicting constraints should be considered carefully.

Soft real-time means in our case response times from $100ms$ up to $3s$ depending on the number of orders. The more orders to schedule, the less response time the algorithm has. Therefore we compare some variations of the scheduling algorithm with the optimal solutions achieved by exhaustive backtracking.

5.1 Formal description of the problem

First we give the basic notation of the scheduling problem.

Definition 1. Let **Time** $= \{ts \mid ts \in I\!N\}$ with the granularity of seconds.

Definition 2. Let **Offices** be a set of locations $(x, y, z) \in I\!R^3$. Each element of **Offices** is related to the physical location in the middle of an office door. **Offices** $\neq \emptyset$ and contains a discrete number of elements.

Definition 3 Event. An event ev is a 2-tuple (o, t) where $o \in$ **Offices**, and $t \in$ **Time**. **Events** is the set of events.

Given the event $ev = (o, t)$ we define the two expressions $T(ev) = t$ and $O(ev) = o$. An event represents the arrival of a robot to an office at time t. Two useful definitions are related to an event:

Definition 4 Order. An order jo is a 3-tuple $(ev_1, ev_2, (tw_1, tw_2))$ with $ev_1, ev_2 \in$ **Events** and tw_1, $tw_2 \in$ **Time**. An order $jo = ((o_1, t_1), (o_2, t_2), (tw_1, tw_2))$ is valid if it fulfills the constraints:

1. $o_1 \neq o_2$
2. $tw_1 \leq t_1 < t_2 \leq tw_2$. **Order** $\neq \emptyset$ is a set of orders[2].

ev_1 is the pickup event and ev_2 the delivery event in the time window $[tw_1, tw_2]$ of an order. An order is also called a job.

Definition 5 Cost. Let $ev_1 = (o_1, t_1)$, $ev_2 = (o_2, t_2) \in$ **Events** with $o_1 \neq o_2$. The cost for going from o_1 to o_2 is $\mathbf{C(ev_1, ev_2)} = (t_2 - t_1)$, with $\mathbf{C}$: **Events** $\times$ **Events** $\longrightarrow$ **Time**.

The Cost function measures the cost for going from the office o_1 to the office o_2 in seconds.

Definition 6 Schedule. Let $OD \subset$ **Order** and $EV \in$ **Events** a set of events. A preemptive schedule is an indexed set of events $SCH_P = (ev_1, ev_2, \ldots, ev_n)$ where $\forall jo = (ev_i, ev_j, [tw_i, tw_j]) \in OD$:

1. $ev_i = (o_i, t_i)$, $ev_j = (o_j, t_j) \in EV$,
2. $i < j$, $i, j \in \{1 \ldots n\}$,
3. $tw_i \leq t_i < t_j \leq tw_j$.

A non-preemptive schedule is an indexed set of events $SCH_{N-P} = (ev_1, ev_2, \ldots, ev_n)$ where $\forall jo = (ev_i, ev_j, [tw_i, tw_j]) \in OD$:

1. $ev_i = (o_i, t_i)$, $ev_j = (o_j, t_j) \in EV$,
2. $j = i + 1$, $i = 1, 3, 5, \ldots, n-1$, $j = 2, 4, 6, \ldots, n$,
3. $tw_i \leq t_i < t_j \leq tw_j$.

The second condition guarantees in the preemptive case that the delivery event occurs sometime after the pickup event. In the non-preemptive case the delivery event must be directly after the pickup event. The third condition ensures that the estimated delivery dates fit into the time window.

[2] The practical definition of an order contains more items: The owner who is responsible for the order, the task to be performed, the object to carry, the status of the order, and some restrictions. These items are not needed in this formal description.

Definition 7 Cost of a schedule. The cost of a schedule $SCH = (ev_1, ev_2, \ldots, ev_n)$ is defined as:

$$C(SCH) = \sum_{i=1}^{n-1} C(ev_i, ev_{i+1}) = \sum_{i=1}^{n-1} (t_{i+1} - t_i)$$

The definition of the cost of a schedule is independent from the preemptive or non-preemptive case. Now we can formulate more precisely our problem which is to find a schedule SCH with minimal costs.

Definition 8 Minimal Schedule. Let OD a set of n/2 orders and EV the related set of events, i.e., $\mid EV \mid = n$ and SCH_{min} a schedule of the events with $\mid SCH_{min} \mid = n$. SCH_{min} is called minimal, if $\forall$ SCH_i of EV with $SCH_i \neq SCH_{min}$ and $\mid SCH_i \mid = n$: $C(SCH_{min}) < C(SCH_i)$.

5.2 Scheduling Algorithms

Suppose we have a set of orders $OD \in$ **Order** with $\mid OD \mid$ = m. Since each order has two events, the number of events in a schedule is $n = 2 * m$. The goal of the following algorithms is to provide a scheduled list of events (arrivals and departures) that accomplish the time requirements for each order. Therefore, some algorithms and the time complexity are given. In general, scheduling of orders is NP-hard so first two optimal algorithms for the preemptive and for the non preemptive case are presented. Next, some approximation algorithms (suboptimal algorithms) are shown. The main goal of the suboptimal algorithms is to minimize the computational costs to fulfill the soft realtime constraint.

Optimal algorithms

The optimal algorithms are those that find an optimal solution, i.e., a schedule with minimal costs for a set of orders. The schedules from the optimal algorithms are the references for the comparison with our proposed algorithms. They are based on exhaustive search among all the possible combinations of events.

Algorithm 1: Non preemptive

Let $IND = 1, \ldots, m$ an index set and $SCH_{N-P} = (ev_{1,1}, ev_{1,2}, \ldots, ev_{m,1}, ev_{m,2})$ a schedule. An optimal non-preemptive algorithm (O-N-PRE) has to generate all permutations of the index set $1, \ldots, m$ to build the schedules, respectively. The number of possible schedules is $SCH^n_{N-P} = m! = \frac{n}{2}!$.

Algorithm 2: Preemptive

Let $IND = 1, \ldots, n$ an index set and $SCH_P = (ev_1, \ldots, ev_n)$ a schedule. An optimal preemptive algorithm (O-PRE) has to generate SCH^n_P valid permutations of the index set $1, \ldots, n$ with $SCH^n_P = \frac{n!}{\sqrt{2^n}}$ (see Appendix for details).

The main problem of the optimal algorithms in dynamic environments is the computation requirements. The whole schedule must be recalculated when a new order is added to the list of orders, e.g., in the non preemptive case for m = 10 orders (20 events) m! = 3.628.800. A Pentium III 400 Mhz needs 7.4 seconds to build all the schedules and to calculate the minimal costs (Table 2). The minor changing constraint can not be fulfilled. In the preemptive case for n = 10 events (= 5 orders) $\mid SCH_P^n \mid = 113400$. A Pentium III 400 Mhz needs 3.1 seconds to build all the schedules and to calculate the minimal costs (Tab. 1). The minor changing constraint can also not be fulfilled.

Approximation algorithms

Next, four different algorithms for a suboptimal solution are described. The main goal of the suboptimal algorithms are to minimize the additional cost of the schedules when adding a new order $jo = (ev_{\phi,1}, ev_{\phi,2}, (tw_{\phi,1}, tw_{\phi,2}))$ to a previously scheduled list of events $SCH = (ev_1, \ldots, ev_m)$. The output is a new schedule $SCH = (ev_1, \ldots, ev_{m+2})$, with the previous events and the new job. Hereinafter, we call this class of algorithms *minimal additional costs (MAC)* algorithms. In the preemptive scheduling case the two events of an order can be inserted at any position in the event list under the constraint that the delivery event must be placed after the pickup event. The following three different strategies are used:

1. *Forward or backward.* The forward / backward scheduling strategy inserts the pickup event at any position in the event list so that the additional cost is minimal. The delivery event is inserted in the event list between one position after the pickup event and the end of the event list. The condition for that position is also to minimize the additional costs. Alternatively, the delivery event is inserted first and then the pickup event between the first position of the event list and the delivery event. Both strategies are nearly equal but produce different schedules.

2. *Forward-backward.* A combination of both strategies. It checks a forward and a backward insertion and takes the minimum for both events.

3. *Complete.* The complete scheduling strategies checks the insertion of the pickup event at each position and the delivery event after the pickup event until the end of the scheduling list. Finally it take that insertion with the minimal additional cost for both events.

The cost for an insertion of an event ev at a position j are be calculated by:

$$f_{SCH}(e, j) = \begin{cases} C(ev, ev_1) & \text{, if } j = 0 \\ C(ev_j, ev) + C(ev, ev_{j+1}) - C(ev_j, ev_{j+1}) & \text{, if } 1 \leq j < m \\ C(ev_m, ev) & \text{, if } j = m \end{cases}$$

where $SCH = (ev_1, \ldots, ev_m)$ is an indexed schedule and $0 \leq j \leq m$. The formula give the cost of an insertion of the event ev at the position j. If j is between two events than we got additional costs to go from ev_j to ev and from ev to ev_{j+1} but we have no more costs for going from ev_j to ev_{j+1}.

Now, a more formal description of the three algorithms is given. Let $jo = (ev_{\phi,1},\ ev_{\phi,2},\ (tw_{\phi,1}, tw_{\phi,2}))$ a new order with the pickup event $ev_{\phi,1}$ and the delivery event $ev_{\phi,2}$. The time interval in which the order has to be processed by the robot is $[tw_{\phi,1}, tw_{\phi,2}]$.

Algorithm 3: Forward MAC (FMAC)

(1) find index j in $SCH_P = ev_1, \ldots, ev_m$ such that
$f_{SCH_P}(ev_{\phi,1}, j)$ is minimal and
$tw_{\phi,1} \leq T(ev_j) + C(ev_j, ev_{\phi,1}) \leq tw_{\phi,2}$

(2) find index k in $SCH'_P = ev_{j+1}, \ldots, ev_m$ such that
$f_{SCH'_P}(ev_{\phi,2}, k)$ is minimal and
$tw_{\phi,1} \leq T(ev_k) + C(ev_k, ev_{\phi,2}) \leq tw_{\phi,2}$

(3) the result is
$SCH''_P = ev_1, \ldots, ev_j, ev_{\phi,1}, ev_{j+1}, \ldots, ev_k, ev_{\phi,2}, ev_{k+1}, \ldots, ev_m$

Algorithm 4: Forward-Backward MAC (FBMAC)

(1) find index j in $SCH_P = ev_1, \ldots, ev_m$ such that
$f_{SCH_P}(ev_{\phi,1}, j)$ is minimal and
$tw_{\phi,1} \leq T(ev_j) + C(ev_j, ev_{\phi,1}) \leq tw_{\phi,2}$

(2) find index k in $SCH'_P = ev_{j+1}, \ldots, ev_m$ such that
$f_{SCH'_P}(ev_{\phi,2}, k)$ is minimal and
$tw_{\phi,1} \leq T(ev_k) + C(ev_k, ev_{\phi,2}) \leq tw_{\phi,2}$

(3) $SCH^f_P = ev_1, \ldots, ev_j, ev_{\phi,1}, ev_{j+1}, \ldots, ev_k, ev_{\phi,2}, ev_{k+1}, \ldots, ev_m$

(4) $cost^f_P = f_{SCH_P}(ev_{\phi,1}, j) + f_{SCH'_P}(ev_{\phi,2}, k)$

(5) find index p in $SCH_P = ev_1, \ldots, ev_m$ such that
$f_{SCH_P}(ev_{\phi,2}, p)$ is minimal and
$tw_{\phi,1} \leq T(ev_p) + C(ev_p, ev_{\phi,2}) \leq tw_{\phi,2}$

(6) find index q in $SCH''_P = ev_1, \ldots, ev_p$ such that
$f_{SCH''_P}(ev_{\phi,1}, q)$ is minimal and
$tw_{\phi,1} \leq T(ev_q) + C(ev_q, ev_{\phi,1}) \leq tw_{\phi,2}$

(7) $SCH^b_P = ev_1, \ldots, ev_q, ev_{\phi,1}, ev_{q+1}, \ldots, ev_p, ev_{\phi,2}, ev_{p+1}, \ldots, ev_m$

(8) $cost^b_P = f_{SCH_P}(ev_{\phi,2}, p) + f_{SCH''_P}(ev_{\phi,1}, q)$

(9) the result is $\begin{cases} SCH^f_P, \text{ if } cost_f \leq cost_b \\ SCH^b_P, \text{ else} \end{cases}$

Algorithm 5: Complete MAC (CMAC)

(1) $\forall j = 0 \ldots m$ and $\forall k = j+1 \ldots m$ compute:
$g_{j,k} = f_{SCH_P}(ev_{\phi,1}, j) + f_{SCH'_P}(ev_{\phi,2}, k)$

(2) the result is
$SCH'_P = ev_1, \ldots, ev_p, ev_{\phi,1}, ev_{p+1}, \ldots, ev_q, ev_{\phi,2}, ev_{q+1}, \ldots, ev_m$
with: $g_{p,q} \leq g_{j,k}$, $j = 1 \ldots m$, $k = j+1 \ldots m$ and
$tw_{\phi,1} \leq T(ev_p) + C(ev_p, ev_{\phi,q}) \leq tw_{\phi,2}$ and
$tw_{\phi,1} \leq T(ev_q) + C(ev_q, ev_{\phi,2}) \leq tw_{\phi,2}$

In the non-preemptive case the delivery event must be directly after the pickup event in the list of events. This condition reduces the number of effective strategies to a modified CMAC algorithm.

Algorithm 6: Non preemptive MAC (MAC-N-PRE)

(1) $\forall j = 0 \ldots m$ compute:

$$g_j = f_{SCH_N - P}(ev_{\phi,1}, j) + f_{SCH'_N - P}(ev_{\phi,2}, j+1)$$

(2) the result is $SCH'_N - P = ev_1, \ldots, ev_k, ev_{\phi,1}, ev_{\phi,2}, ev_{k+1}, \ldots, ev_m$

with: $g_k \leq g_j$, $j = 1 \ldots m$ and

$$tw_{\phi,1} \leq T(ev_k) + C(ev_k, ev_{\phi,q}) + C(ev_{k+1}, ev_{\phi,2}) \leq tw_{\phi,2}$$

All of the above approximation algorithms minimizes the cost function only locally and can not find the global optimum in any case but the minor changing constraint from section 1 is fulfilled.

Note: Empty list of schedules are not considered in the description of the algorithms. Since each robot has a current position this position is the start event and the last event is the return to the depot. They will be always the first and the last event and will not have a schedule time.

We formulate the scheduling algorithms for one service robot. The extension to multiple robots is obvious, i.e., each robot has his own scheduling queue and the scheduling is tested for each robot and scheduled to the robot with the minimal additional costs. Two strategies called “equal balanced” and “non-equal balanced” are distinguished. Equal balanced means that all of the robots have nearly the same number of orders whereas in the non-equal case these number is highly different. Results will be given in the next section

6 Implementation and results

All algorithms have been implemented in C. For the comparison we suppose that all time intervals are equal and long enough. All time values are given for a Pentium III 400 Mhz. Two kinds of values have to be considered. The first one is the cost of a schedule which is the time in seconds a robot needs to execute the schedule and the second the computation time required for generating the schedule. The following tables summarize the results. In each table four columns for each algorithms are given. First, the name of the algorithm. Next, the average and worst difference to the best solution (schedule costs) and last the computation time needed to build the schedule on a Pentium III. Since the order of the orders is important for the schedules, random list of orders are generated in 1000 experiments, scheduled and statistically analyzed.

Table 1 shows the summarize results for 5 orders (= 10 events) which is the practical maximum number of orders for the optimal algorithms in the preemptive case. The table shows that the average difference between the preemptive and the non-preemptive case is around 32%. So, for robots it is important to have a high load capacity to increase throughput.

Alg.	average diff. %	worst diff. %	calculation time (ms)
Randomly	47.74	250.00	0.13
MAC-N-PRE	33.01	204.30	0.25
O-N-PRE	31.50	204.30	1.67
FMAC	8.12	59.00	0.17
FBMAC	4.76	49.30	0.23
CMAC	4.08	50.10	0.33
O-PRE	0.00	0.00	3068.96

Table 1: Comparison of the scheduling algorithms. In 1000 cycles 5 orders are selected randomly and scheduled.

Alg.	average diff. %	worst diff. %	calculation time (ms)
Randomly	51.40	276.60	0.22
MAC-N-PRE	8.14	48.70	0.48
O-N-PRE	0.00	0.00	7404.74

Table 2: Comparison of the different non-preemptive scheduling algorithms. In 1000 cycles 10 orders are selected randomly and scheduled.

Table 1 also shows that the time consumption for the suboptimal algorithms is low but the time difference (costs) in the worst case could be around 50%. The large value of the computation time for optimal preemptive case is related to combinatorial explosion and the exhaustive nature of the algorithm. The scheduling result in the preemptive case is the reference for all other algorithms.

The next two tables 2 and 3 show the results for the non-preemptive and preemptive algorithms separately. Table 2 shows the summarize scheduling results for 1000 trials and 10 orders. Regarded to the computation time, 10 orders are the practical maximal number of orders which could be scheduled from the optimal non-preemptive algorithm. The approximation algorithms in table 2 is much faster than the optimal but could lead to 50% more traveling costs of the robot. On the other hand a random insertion algorithm, e.g., first come, first serve with the cheapest computation time (nearly zero) leads really worst schedules.

The last table 3 shows the results of the different approximation algorithms for 100 orders. Since no optimal solution is available in table 3 (for 100 orders), the average and worst difference between the schedules found by the algorithms and the know best solution is given.

Alg.	average diff. %	worst diff. %	calculation time (ms)
FMAC	22.53	122.90	15.26
FBMAC	5.68	62.90	28.20
CMAC	1.01	18.50	972.55

Table 3: Comparison of the three different suboptimal scheduling algorithms. In 1000 cycles 100 orders are selected randomly and scheduled. Since the optimal solution is not available the difference are given to the best known schedule.

As expected, the CMAC algorithm is the best preemptive approximation algorithm but with higher computation requirements then the FMBAC or FMAC algorithms. FBMAC obtains better results than FMAC and similar to CMAC but with double computational effort than FMAC. Figure 9 left shows the scheduling costs of the preemptive approximation algorithms for different number of orders. Each order number (5 - 100) is simulated and summarized 1000 times.

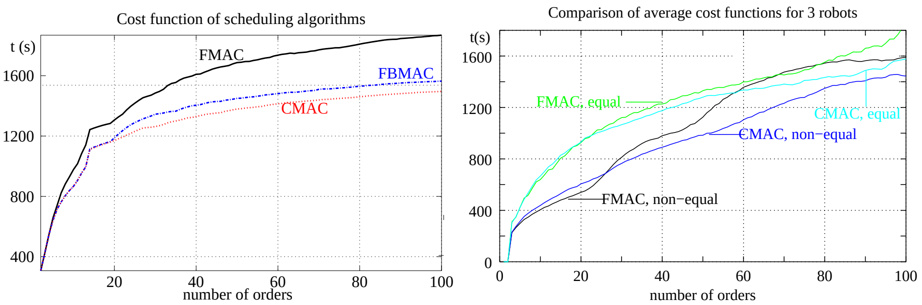


Figure 9: Left: Scheduling costs of the preemptive algorithms for 1-100 orders and one robot. Right: Average scheduling costs of the FMAC and CMAC algorithms for three robots (equal balanced / non equal balanced) and 100 jobs per robot

Figure 9:right and figure 10 show some results for multiple robots. Each of the robot has its own list of events whereas the scheduler schedules new orders to these different lists of events. Two different strategies are compared called equal balanced and non equal balanced. In the equal balanced case the scheduler schedules the job to the event queue of the robot with the lowest number of events. If multiple events are in the event list a MAC algorithm is used to insert

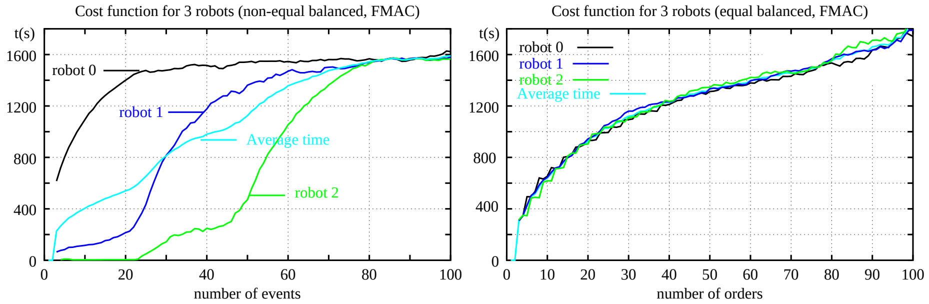


Figure 10: Scheduling costs of the FMAC algorithm for three robots. Left: non equal balanced; Right: Equal balanced (100 jobs per robot).

the new events. Figure 10:right show that the cost function is nearly equal for all of the robots. In the non equal case a new order is scheduled to the list of events with the global minimal additional costs of all robot queues. Figure 10:left shows that one robot is loaded after each other. This behavior results from the fact that a new order generates lower costs if a robot has a tour to the buildings compared to a new robot coming out of the robot depot. This change when the order could not be scheduled in the time interval which leads to additional costs (punishment). Then it is cheaper to take a new robot. Figure 9:right compares the two different approximation algorithms CMAC and FMAC according to the equal and non equal strategy. It shows the average cost functions for all of the three robots. The non equal distribution strategy generates lower average costs especially for lower number of jobs. The distribution strategy has a higher impact to cost function than the approximation algorithm. If all robots operate at nearly full capacity the quality of the approximation algorithms mainly determines the costs and the CMAC algorithm is the best.

6.1 Implementation details

We presented above some definitions and formal descriptions of the algorithms. It is not difficult to develop practical implementations from these descriptions, but some practical aspects have to be considered.

In the model the physical location of an office is the location of its doors, since this is the point that should be reached by the mobile robots. Due to the structure of the environment the doors are grouped to corridors and corridors to floors. All the floors are connected by elevators. The model of the environment influences the schedule by means of the cost function. Indeed, a detailed description of the model is the base of the path planner at the higher abstraction levels. The planner generates appropriate paths from pickup office to the delivery office.

The cost function of definition 5 is only a value that represents the cost of going from one office to another one. In our case this costs are measured in units of seconds. From the practical point of view it is very important to have an accurate estimation of some parameters (see below) in order to obtain a good performance of the algorithms. This accuracy has been obtained following an adjusting process. The average of the time values observed in the real operation of the robots has been used to build the estimations. The important average values are:
The **average speed of the robots** going through the corridor, the **average time to open a fire protection door** at the end of the corridor and the **average time using the elevator** and moving to different floors. Further parameter are the average stopping time for loading / unloading the robot. These values are monitored from the server and updated once a day. The scheduler adapts/learn the working environment while office environments are different. The schedule could be fine tuned if an array of adaptation values is used, e.g., for different corridors. Nevertheless, the time estimation with only one average value is quite good while the estimations are updated always if an event occurs, e.g., after a pickup or a delivery event or when a new order is entered. Furthermore the unprocessed orders are rescheduled at regular time intervals, e.g., every 5 minutes and updated with current time values to reach an actual time estimation especially when unexpected problems occur.

The scheduling strategy for multiple robots, i.e., having a scheduling queue for each robot and selecting the queue with the minimal additional cost leads to load one robot after each other. Unused robots are waiting in the depot and usually the scheduling costs from the depot to two offices are higher than for a moving robot. Additional costs are added to a schedule if a time window fails. If one robot could not fulfill an order in time, i.e., a schedule runs out of its time window then the order will be scheduled to the next roboter because of the extra costs (non equal balanced strategy).

7 Conclusion

We have presented a general approach for scheduling orders for a fleet of autonomous mobile service robots in indoor environments. Each of the autonomous mobile robots has a queue of events and new orders will be scheduled to the robots queues. It has been developed, implemented and validated in the framework of the finished ARIADNE project. An online version is available at http://-lamu.gmd.de:8080.

The multi robot cooperation, which is one key problem, has been studied by several scheduling algorithms and by giving *autonomy* to the service robots. We have presented the scheduling algorithms, their quality and timing capabilities. The quality of the schedules depends on the number of orders and the calculation time. Furthermore a customer friendly web based interface and a general language for modeling multistorey buildings was also presented.

The main feature of our approach is the combination of nearly optimal routes in terms of time and space and customer-orientation. Customer-orientation requires a customer friendly interface and it takes into account all the needs of users like privacy, online capability, up-to-date scheduling information, planning certainty, time estimations and real-time response. The soft real-time response is achieved by cost effective insertions of new orders in the order lists of the robots and guarantees good routes and the online running of the whole system.

Acknowledgement
The authors would like to thank Daum & Partner GmbH, Schmersal EOT, the members of the ARIADNE team, Joachim Hertzberg and Claudia Hirsch for supporting this work. The second author is supported in part by the Ministerio de Ciéncia y Tecnologia of Spain under a FPI Program graduate fellowship, by the CICYT under project DPI2001-3801, by the Conselleria de Cultura i Educació (Generalitat Valenciana) under project GV01-244, and by the Fundació Caixa-Castelló under project P1.

References

[Ark98] Ronald C. Arkin. *Behavior-based Robotics*. The MIT Press, Cambridge, Massachusetts, London, England, 1998.

[BCF+98] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Gerhard Lakemeyer, Dirk Hähnel, Dirk Schulz, Walter Steiner, and Sebastian Thrun. The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[BF96] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(5):869–880, 1996.

[Cor98] Netscape Communication Corp. Ssl version 3.0. *http://-home.netscape.com/eng/ssl3/index.html*, 1998.

[DG96] John December and Mark Ginsburg. *HTML and CGI unleashed*. SAMS net. pub., sep 1996.

[Eng93] Joseph F. Engelberger. Health-care robotics goes commercial: the ‘Help-Mate’ experience. *Robotica*, 11(VII):517–523, March 1993.

[HV97] Karen Zita Haigh and Manuela M. Veloso. High-level planning and low-level execution: Towards a complete robotic agent. In W. Lewis Johnson, editor, *Proceedings of First International Conference on Autonomous Agents*, pages 363–370, Marina del Rey, CA, February 1997. ACM Press, New York, NY.

[Inc98] Open Market Inc. The unofficial fastcgi homepage. *http://-www.fastcgi.com*, 1998.

[KS96] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAi-96). Portland Or.*, 1996.

[Lau98] Ben Laurie. Apache-ssl. *http://www.apache-ssl.org, http://-www.links.org/homes/benhome.html*, 1998.

[LBB83] A. Assad L. Bodin, B. Golden and M. Ball. The routing and scheduling of vehicles and crews – the state of the art. *Computers and Operations Research*, 10(2):66–212, 1983.

[LN87] G. Laporte and Y. Novert. Exact algorithms for the vehicle routing problems. *Annals of Discrets Mathematics*, 31:147–184, 1987.

[Mag81] T. Magnanti. Combinatorial optimazation and vehicle fleet planning: Perspectives and prospects. *Network*, 11:179–214, 1981.

[PSFL98] Michael Pauly, Hartmut Surmann, Marion Finke, and Nang Liang. Real-time object detection for autonomous robots. *Informatik Aktuell. Autonome Mobile Systeme, 14. Fachgespräch, Springer-Verlag*, pages 57–64, 1998.

[RAHIR98] S. Fleury R. Alami, M. Herrb, F. Ingrand, and F. Robert. Multi robot cooperation in the martha project (http://www.laas.fr/∼felix/publis/ieee-ram96/paper.html). *IEEE Robotics and Automation Magazine Special Issue on "Robotics & Automation in the European Union"*, 5(1), Mar. 1998.

[RG98] P. Weckesser R. Graf. Roomservice in a hotel. In *IFAC Symposium on Intelligent Autonomous Vehicle, Madrid*, pages 641–647, 1998.

[Saf97] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation (http://aass.oru.se/agora/flar/survey/). *Soft Computing*, 4(1):180–197, 1997.

[SD88] M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13, 1988.

[SGH+97] Reid Simmons, Rich Goodwin, Karen Zita Haigh, Sven Koenig, and Joseph O'Sullivan. A layered architecture for office delivery robots. In W. Lewis Johnson, editor, *Proceedings of First International Conference on Autonomous Agents*, pages 245–252, Marina del Rey, CA, February 1997. ACM Press, New York, NY.

[SHP95] Hartmut Surmann, Jörg Huser, and Liliane Peters. A fuzzy system for indoor mobile robot navigation. In *Fourth[SHP95] IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 95, Yokohama, Japan*, pages 83–88, 20–24 March 1995. Distinguished with *Robot Intelligence Award*.

[SHW96] Hartmut Surmann, Jörg Huser, and Jens Wehking. Path planning for a fuzzy controlled autonomous mobile robot. In *Fivth International Conference on Fuzzy Systems, FUZZ-IEEE 96, New Orleans, USA*, pages 1660 – 1665, September 1996.

[SM00] Hartmut Surmann and Antonio Morales. A five layer sensor architecture for autonomous robots in indoor environments. In *International symposium on robotics and automation ISRA'2000, Monterrey, N.L., Mexico*, pages 533–538, 2000.

[SP01] Hartmut Surmann and Liliane Peters. *MORIA - A Robot with Fuzzy Controlled Behaviour*, volume 61 of *Studies in Fuzziness and Soft Computing*, chapter Layer Integration, pages 343–365. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 2001.

[ST99] Hartmut Surmann and Markus Theißinger. Robodis: A dispatching system for multiple autonomous service robots. In *Proceedings of the FSR'99, Robotics Applications for the Next Millenium, Pittsburgh, PE*, pages 168 – 173, 29.8. - 31.8.1999.

[SV96] R.D. Schraft and H. Volz. *Serviceroboter, Innovative Technik in Dienstleistung und Versorgung*. Springer-Verlag, Berlin, Heidelberg, 1996.

[SW00] Petra Schürman and Gerhard J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *SIAM Journal on Computing*, page to appear, 2000.

[TBB+99] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremmers, F. Dellaert, D. Fox, D. Hähnel, G. Resmeberg, N. Roy, J. Schulte, and D. Schultz. Minerva: A second generation mobile tour-guide robot. In *Proc. of the IEEE International Conference on Robotics & Automation (1999)*, 1999.

[TW86] M. Togai and H. Watanabe. An inferecne engine for real-time approximate reasoning: Toward an expert on a chip. In *IEEE EXPERT*, volume 1, Nr. 3, pages 55 – 62, 08/1986.

[Vep99] Linas Vepstas. Raid solutions for linux. *http://linas.org/linux/raid.html*, 1999.

[Ves96] S.J. Vestli. Mops - a system for mail distribution in office-type buildings. *Service Robot: An International Journal*, 2(2):29–35, 1996.

[YDS91] J. Desrosiers Y. Dumas and F. Suomis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 1(54):7, 1991.

A APPENDIX

Let $P_n = \{\sigma | \sigma$ a permutation over $n\}$ then the number of permutation or the size of $|P_n| = n!$

For $i < n$ let $pos_\sigma(i)$ be a function with $pos_\sigma(i)$ denotes the position of i in $\sigma(1, \ldots, n)$ e.g. $\sigma(1,2,3,4) = (3,1,2,4)$ then $pos_\sigma(1) = 2$, $pos_\sigma(2) = 3$, $pos_\sigma(3) = 1$, $pos_\sigma(4) = 4$

Let n even e.g. $(n = 2, 4, 6, \ldots)$ and $S_n = \{\sigma | \ \forall$ odd i $(i = 1, 3, 5, \ldots) : pos_\sigma(i) < pos_\sigma(i+1)\}$ then

Lemma 1 *The number of the schedules* $SCH_n = \frac{n!}{\sqrt{2^n}}$ *in the preemptive case*

Note:
The pickup event must precede its delivery event, so only a half of the schedules make semantic sense. The rest are discarded.

Proof induction over all even n $(n = 2, 4, \ldots)$

For n=2: $SCH_2 = \frac{2!}{\sqrt{2^2}} = 1$ and $P_n = \{\sigma_1 = (1,2)$ and $\sigma_2(2,1)\}$ with $|P_n| = 2$. $\sigma_1 \in SCH_2$: $pos_{\sigma_1} = 1 < pos_{\sigma_2} = 2$ $\sigma_2 \notin S_2$: $pos_{\sigma_2} = 2 \geq pos_{\sigma_1} = 1$. So $S_2 = \{\sigma_1\}$ and $SCH_2 = 1$.

$n \rightarrow n+2$ For n is: $SCH_n = \frac{n!}{\sqrt{2^n}}$. Now, two new elements have to be added. The first one can be placed at any position, i.e., we will have $n+1$ possibilities. Than for the second event we have $n+2$ possibilities to place in the schedule. If $\sigma \in SCH_n$ e.g. for all odd $i = 1, 3, \ldots$: $\sigma(i) < pos_\sigma(i+1)$
Therefore, we have (n+1)(n+2) possibilities to insert n+1 and n+2 but only half of them are valid e.g. $pos(n+1) < pos(n+2)$ (because of the symmetry).

$$\begin{aligned} So, \quad SCH_{n+2} &= SCH_n * (n+1)(n+2)/2 \\ &= \frac{n!}{\sqrt{2^n}} * \frac{(n+1)*(n+2)}{2} \\ &= \frac{(n+2)!}{\sqrt{2^n} * \sqrt{2^2}} = \frac{(n+2)!}{\sqrt{2^{n+2}}} \end{aligned} \tag{1}$$