Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Matrix-based scheduling and control of a mobile sensor network

V. Schiraldi*, V. Giordano*, D. Naso*, B. Turchiano*, F. L. Lewis**

* Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari,
Bari 70125, Italy. (email naso@poliba.it)
**Automation and Robotics Research Institute, University of Texas at Arlington,
J. Newell Blvd. 7300, Fort Worth, TX 76118 USA. (email lewis@uta.edu).

Abstract: This paper considers real-time coordination of a mobile sensor network composed of heterogeneous resources with partially overlapping functionalities in charge of executing multiple sequences of interconnected tasks. A discrete event controller based on a matrix-based formalism is adopted to combine in a single framework task planning, dynamic resource assignment with look-ahead, and shared resource conflict resolution with utility-based method. The matrix based controller is modular and can be easily reconfigured if mission characteristics or network topology change. Simulations and preliminary results on an experimental platform are provided to illustrate the main features of the proposed control approach.

1. INTRODUCTION

A Mobile Sensor Network (MSN) is a network of mobile, heterogeneous devices that are able to perform a variety of tasks, including measuring, observing, tracking, finding or manipulating objects. Many MSNs use very sophisticated robots as single units, and borders between research areas such as mobile robotics (Gerkey and Mataric, 2004), multi-robot coordination/cooperation (Lee et al., 2003, Burgard, et al., 2005, Tsalatsanis et. al., 2006), and sensor networks (Ren et al., 2006, Dharne et al., 2006) are rapidly vanishing. Mobility and multifunctionality are generally adopted as a means to reduce the number of nodes, at the cost of making the overall supervision and control task much more challenging. In fact, in this case, the control system of the MSN must simultaneously address task planning, dynamic resource assignment, resolution of conflicts for shared resources, and event-based feedback control. On the one hand, it is well-known that all of these sub-problems have been extensively investigated in related areas such as operation research and manufacturing system control. On the other hand, it can be still observed that the preponderance of the related literature focuses on one aspect only, disregarding (or strongly simplifying) the interactions with the other ones. Therefore, recent researches (Giordano et al., 2006) have also remarked that the effective exploitation of the capabilities of multifunctional MSNs calls for integrated approaches capable to properly describe and address all the problems within a unified modelling and decision-making approach.

Recent research has identified the "*Matrix-based Discrete Event Control*" (M-DEC) formalism (Tacconi and Lewis, 1997) as an effective tool to address this problem. This discrete-event modelling technique has been applied to a variety of complex, large-scale distributed systems, e.g. (Mireles and Lewis, 2002). In (Giordano et al., 2006) the M-DEC is used to control a MSN for which it is assumed that task allocation is predetermined offline by decision-makers using a priori information. Since many MSN need to cope with (1) multiple competing missions issued at unpredictable times, and (2) dynamic network topologies deriving from mobility, faults and addition/removal of sensors, the integration of a dynamic resource assignment tool in the M-DEC framework is a natural and significant extension of the work presented in (Giordano et al., 2006).

Thus, in this paper the matrix-based DEC is not only in charge of implementing preconditions, postconditions, and interdependencies of the tasks and shared resource management, but it also dynamically determines task-to-resource assignment. Considering the inherently turbulent nature of the MSN operation, we develop a task allocation algorithm that exploits estimates of task durations (when available) to improve the overall task allocation by means of "look-ahead" optimization. The algorithm is integrated within the M-DEC system so as to obtain a single, global description of the closed-loop control system of the MSN. Simulations results for a large MSN with up to 20 mobile sensors are provided and analyzed to evaluate the effectiveness of the proposed strategy. This paper also overviews the implementation of a laboratory-scale MSN with three mobile robots, videocameras and other sensors. The reminder of the paper is organized as follows. Sections 2 and 3 overview the main assumptions and the key-elements of the M-DEC, respectively. Section 4 describes the task assignment algorithm, and Section 5 summarizes the simulation results. Section 6 describes the MSN prototype and section 7 concludes the paper with final remarks.

2. MAIN ASSUMPTIONS

Consider a MSN composed of *q resources* (mobile, transportable or stationary sensors or group of sensors, hereafter simply called *robots*). Each robot has one or more *functionalities*, and the total number of different functionalities available in the MSN is *s*. The MSN has to accomplish a set of *n missions.* Each mission consists of a predefined set of *tasks*, which may be subject to precedence constraints (the end of one task may be a necessary prerequisite for the start of some other ones), while each task needs exactly one functionality. For each couple *(t,p),* where t is a task and p is a robot, a predefined real-valued non-negative *Utility function U(t,p)* is available to describe the desirability or effectiveness of the assignment of task t to resource p. Further, we make the following assumptions:

- The robots may be heterogeneous (non-identical).
- Missions are issued at unknown times, while the number and type of tasks composing a mission is known a priori.
- The communication between robots and M-DEC is based on a wireless system, whose failures, congestion or limitations will not be considered.

For scheduling purposes, it must be also considered that the speed of execution, or more generally the time needed by two or more robots to perform a given task, may vary (one robot may be faster, or closer to the target, than the other ones). In our framework, we assume that some tasks may have completely unpredictable execution times (e.g., finding a lost object), while other ones have a known duration which depends on the particular task-to-robot assignment. Following the taxonomy proposed in (Gerkey and Mataric, 2004), our scenario is an instance of the *Single-Task* (ST) robot *Single-Robot* (SR) task case (ST-SR). The taxonomy classifies MSN problems also according to the nature of the information available for task assignment. Since we consider a turbulent and uncertain environment in which the available information does not permit offline scheduling of robots and tasks, our scenario falls in the class of Instantaneous Assignment (IA) problems. However, it can also be noted that the use of look-ahead evaluation in the task assignment algorithm can be viewed as a particular form of short-term planning.

3. OVERVIEW OF THE MATRIX-BASED DEC

The overall architecture of the M-DEC control system is the same adopted in (Giordano et al., 2006). Briefly, the M-DEC is a centralized supervision system which receives information about new missions as an external input, assigns the tasks to the robots and controls their execution, taking into account their priorities and synchronizing the activities accordingly. The M-DEC provides a rigorous, yet intuitive mathematical framework to represent the mission planning of a multi-robot system according to linguistic if-then rules as:

Rule i: If <*robot 1* has completed *task 3* of *mission 5* and *robot 2* is available> then <*robot 2* starts *task 4* of *mission 5* and *robot 1* starts *task 3* of *mission 2*>.

In particular within the matrix based formalism it is possible (1) to model the MSN discrete event dynamics and (2) to implement supervisory control policies to define resource assignment, conflict resolution and task priorities.

3.1. Discrete-event system model

Let us consider a MSN of q robots in charge of performing n missions, each one triggered by a predefined event and composed of a predefined sequence of tasks fired when certain logical conditions are met. In the matrix-based modelling formalism, the value of the logical conditions for the activations of q rules of a certain mission i composed of p tasks is summarized by the *rule logical vector*

$$x^i = \begin{bmatrix} x^i(1) & x^i(2) & \dots & x^i(q) \end{bmatrix}^T \tag{1}$$

An entry of '1' in the j-th position of vector x^i denotes that rule j is currently fired (all the preconditions are true).

The conditions of the tasks of each active mission are described by two vectors, the *mission task-in-progress vector* v_{IP}^i, and the *mission task-completed vector* v^i:

$$v_{IP}^i = \begin{bmatrix} v_{IP}^i(1) & v_{IP}^i(2) & \dots & v_{IP}^i(p) \end{bmatrix}^T \tag{2}$$

$$v^i = \begin{bmatrix} v^i(1) & v^i(2) & \dots & v^i(p) \end{bmatrix}^T \tag{3}$$

Each component of v_{IP}^i (v^i) is set to "1" when the corresponding task is in-progress (completed), and "0" otherwise. In particular, vector v^i represents a fundamental precondition of many logical rules of the matrix-based model, while vector v_{IP}^i is mainly necessary to complete the characterization of each mission (useful for the purpose of computer simulation). Hereinafter, for brevity, we will focus on v^i only, and omit the further details about v_{PI}^i and its dynamical update rules.

Similarly, let us indicate with u^i and y^i, the Boolean input and output variables of mission i, respectively. Variable u^i is set to "1" when the triggering event (fire alarm, intrusion, etc.) of mission i has occurred, and y^i is set to "1" when mission i is completed. Finally let us define the Boolean resource vector r having q elements where an entry of '1' represents 'resource currently available'. In order to model n simultaneous MSN missions we define the *global task vector*

$$\underline{v} = \begin{bmatrix} (v^1)^T & (v^2)^T & \dots & (v^n)^T \end{bmatrix}^T, \tag{4}$$

obtained by stacking the n column task vectors. Similarly, we can define the *global input vector*

$$\underline{u} = \begin{bmatrix} (u^1)^T & (u^2)^T & \dots & (u^n)^T \end{bmatrix}^T, \tag{5}$$

the *global output vector*

$$\underline{y} = \begin{bmatrix} (y^1)^T & (y^2)^T & \dots & (y^n)^T \end{bmatrix}^T, \tag{6}$$

and the *global logical vector*

$$\underline{x} = \begin{bmatrix} (x^1)^T & (x^2)^T & \dots & (x^n)^T \end{bmatrix}^T. \tag{7}$$

Vectors $\underline{v}$, $\underline{u}$, $\underline{y}$ and $\underline{x}$ have a dynamic size depending on the number of tasks of each of the missions currently in progress. In particular, at each time a new mission is activated due to external circumstances, the vectors are resized accordingly, introducing new elements with appropriate values (e.g., the vector u is updated adding a new element set to "1").

All these vectors provide a static description of the conditions of the MSN at a given time. In the following we illustrate the update laws defining the evolution of the system variables over time. The model is run in discrete-time, i.e. the effects of events are computed at the first sample time after their occurrence. When no event occurs between two sample times, all the system variables remain unchanged. Hereafter, except where noted differently, all matrix operations are defined in the or/and algebra, where $\vee$ denotes logical OR, $\wedge$ denotes logical AND, $\oplus$ denotes logical XOR, and overbar indicates negation [see (Tacconi and Lewis, 1997) for an introduction].

Before the description of the update rules, it is convenient to introduce some fundamental matrices. Let F_v^i be the task sequencing matrix for an individual mission. F_v^i has element (j,h) set to "1" if the completion of task $v^i(h)$ is a necessary prerequisite for rule logical vector $x^i(j)$. Similarly, let F_r^i be the resource requirements matrix having element (j,h) set to "1" if the condition of *h-th* resource $\underline{r}(l)$ is an immediate prerequisite for rule logical vector $x^i(j)$. Moreover, let F_u^i be the input matrix, having element (j,h) set to "1" if the occurrence of input *h-th* is an immediate prerequisite for rule logical vector $x_i(j)$.

As for the vectors of variables, the matrices related to the list of n missions can be easily obtained by either stacking or concatenating together the matrix blocks corresponding to each individual mission. As an example, we report the task sequencing matrix F_v for n missions

$$F_v = diagblock\left(F_v^1, F_v^2, \ldots, F_v^n\right). \tag{8}$$

The update of vectors are computed according to the following equations.

After a task starts the corresponding resource becomes busy and the corresponding element of vector r is set to 0:

$$r=r-F_r^T \wedge x \tag{9}$$

(the "minus" operator applied to Boolean components should be interpreted as an XOR operation). At the end of a task, the controller will command the release of the corresponding robot and the MSN feeds the "resource released" input, which is directly used in the model to reset the value of r to the idle ("1") condition.

The update of vector $\underline{v}$ occurs in two different cases. The first one is when the M-DEC model receives the "*j-th* task of *i-th* mission completed" message from the MSN. In such a case, the corresponding element of $\underline{v}$ is set to one ($v_i(j)=1$). The second case occurs when it is set to "0" the element of vector corresponding to previously completed tasks using the following equation

$$\underline{v} = \underline{v} \oplus \left(F_v\right)^T \wedge \underline{x}. \tag{10}$$

Mission inputs and outputs vectors are updated with similar equations (Tacconi and Lewis, 1997).

3.2. Supervisory controller

At the supervisory level, the main function of the M-DEC is to determine which rules must be fired, which tasks must be started, and which resources are in charge of performing the tasks. These functions are processed by means of two different sets of logical equations, one for checking the conditions for the activation of a generic rule of a generic mission, and one for defining the consequent controller outputs. The updated value of the rule logical vector is computed with the following controller state equation

$$\overline{\underline{x}} = (F_v \wedge \overline{\underline{v}}) \vee (F_r \wedge \overline{\underline{r}}) \vee (F_u \wedge \overline{\underline{u}}) \vee (F_{u_d} \wedge \overline{\underline{u}}_d). \tag{11}$$

Matrix F_{u_d} in equation (11) is called the MSN conflict resolution matrix and is used to model the influence of control input $\underline{u}_d$ on the rule vector $\underline{x}$. In particular, an entry of "1" ("0") in $\underline{u}_d$ disinhibits (inhibits) the activation of the corresponding task. The vector $\underline{u}_d$ can be seen as one controllable input of the MSN. Depending on the way one selects the strategy to assign the control vector $\underline{u}_d$, dispatching decisions can be implemented. Also, by dynamically updating matrices F_r *(*which basically defines which resource performs each task), different resource to task assignments can be performed.

On the ground of the current value of the rule logical vector x, the controller determines which tasks to start and which resources to release by means of the matrix controller output equations. In particular, the command of a task start is performed by means of the task start vector $\underline{v}_s$ (having the same structure of vectors v and v_p) using the following Boolean matrix equation

$$\underline{v}_s = S_v \wedge \underline{x}, \tag{12}$$

where S_v is the task start matrix and has element (j,h) set to "1" if logic state $x(h)$ triggers the start command $v_s(j)$ for the corresponding task. Similarly, the resource release command is performed by means of the Boolean MSN resource release vector r_s using the following matrix equation

$$r_s=S_r x. \tag{13}$$

in which S_r is the resource release matrix, having element (j,h) set to "1" if logic state $x(h)$ triggers the release command for the j-th nodes of the MSN. Therefore, vector r_s has q elements, each set to "1" when the controller commands the release of the corresponding MSN unit.

In typical operating conditions, each MSN has several active missions, and multiple tasks waiting for execution. The next section presents the policies used to select, at any given time, the most appropriate tasks and resources in order to avoid conflicts and meet operational requirements. In particular, at each sample time:

- the dispatcher (see next section) implements task conflict resolution and task priorities (specified by u_d *and* F_{ud}) and resource assignment (specified by F_r),

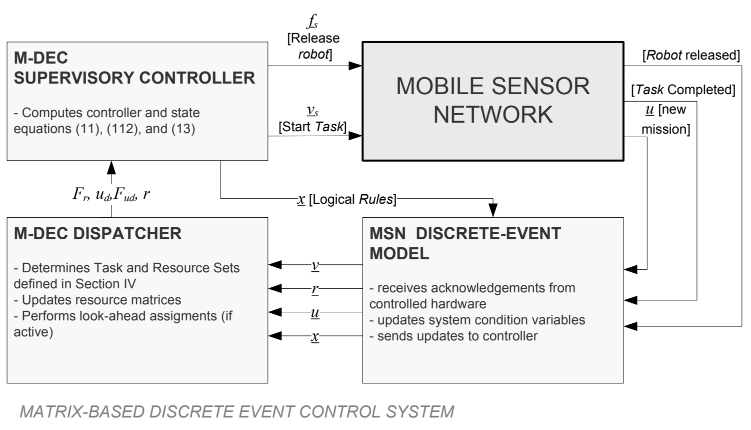
M-DEC SUPERVISORY CONTROLLER
- Computes controller and state equations (11), (112), and (13)
$\underline{f}_s$ [Release robot]
$\underline{v}_s$ [Start Task]
MOBILE SENSOR NETWORK
[Robot released]
[Task Completed]
$\underline{u}$ [new mission]
F_r, u_d, F_{ud}, r
$\underline{x}$ [Logical Rules]
M-DEC DISPATCHER
- Determines Task and Resource Sets defined in Section IV
- Updates resource matrices
- Performs look-ahead assigments (if active)
$\underline{v}$
$\underline{r}$
$\underline{u}$
$\underline{x}$
MSN DISCRETE-EVENT MODEL
- receives acknowledgements from controlled hardware
- updates system condition variables
- sends updates to controller
MATRIX-BASED DISCRETE EVENT CONTROL SYSTEM

Fig. 1. MSN: General architecture of the M-DEC, including the discrete event model, the dispatcher and the supervisory controller

- the controller sends to the MSN the task start and the resource release commands specified by vectors $\underline{v}_s$ and r_s respectively.

Fig.1 illustrates the architecture of the M-DEC, including the new task assignment module described in the next section.

4. TASK ASSIGNMENT

(Gerkey and Mataric, 2004) suggest that ST-SR-IA problems can be viewed as *Optimal Assignment Problems* (OAP), and solved with various centralized or distributed methods (Gerkey and Mataric, 2004). Among the possible methods, the assignment strategy used in this paper is inspired to the "*Min Conflict with Happiness*" (MCH) heuristics proposed in (Gage and Murphy, 2004). The MCH is an algorithm for IA in a preemptive environment, and can be summarized as a sequence of three steps: a first *greedy search* for an initial (feasible or unfeasible) assignment, followed by two different (*local and global*) repair procedures which eliminate, as much as possible, any unfeasible assignment generated in the first step. Our assignment algorithm is structured in the same sequence of three steps, but the procedures used in each step are significantly different. The algorithm works dynamically grouping tasks and resources in a number of sets, updated at each sampling time τ_{now} of the M-DEC. On the ground of the M-DEC discrete event model variables, the dispatcher is essentially in charge of updating matrices F_r, F_{ud} and vectors u_d and r, and of feeding them to the supervisory controller. In particular, for this purpose, the dispatcher calculates all the following sets (which can be easily derived from M-DEC model by means of simple matrix operations that are omitted for brevity):

- T_{CO} *(completed tasks).* This is the set of completed tasks, which is obviously disregarded by the scheduling procedures.
- T_{IP} *(tasks in progress).* This is the set of the task in execution at time τ_{now}(represented by v_p in the M-DEC). As pre-emption is not allowed, also these tasks are not subject to scheduling decision. However, if their expected completion time is known, such information is used for pre-scheduling of other activities.
- T_{PA} *pre-assigned tasks.* This set includes all the tasks waiting for process (i.e. not yet started) but already pre-assigned to resources by look-ahead assignment. For the assignment algorithm, the tasks in T_{PA} are considered equivalent to those in T_{IP} (i.e., decisions about preassigned tasks cannot be changed).
- T_{NE} *non-executable.* This set encompasses all the tasks that cannot be started at time τ_{now} due to precedence constraints with other tasks (e.g., "measure temperature at location x_0" needs task "reach location x_0" to be in T_{CO} before activation).
- T_{UA} *un-assigned tasks.* This set includes all the remaining tasks, which are not assigned to resources but ready for execution at time τ_{now}.

- P_{ID} *(idle resources).* This set includes all the resources idle at time τ_{now}. (described by "1" elements of r in the MDEC).
- P_{PA} (*pre-assigned resources*). This is the set of resources for which the following two conditions simultaneously occur: (1) at time τ_{now}, the resource is processing one task in T_{IP} which is expected to end before $\tau_{now}+\Delta\tau$, where $\Delta\tau$ is the width of the look-ahead time window; (2) the resource has already been preassigned to a new task in T_{PA} by a previous iteration of the look-ahead algorithm.
- P_{EJ} (*resources ending job*). This is the set of resources for which only the first one of the conditions specified for P_{PA} is true. In other words, this set includes all the resources that are available for pre-assignment at time τ_{now}.
- P_{BU} *(busy resources).* This is the set of resources that are not included in any of the sets defined above. These resources are currently processing a task whose expected end is either unknown or exceeds the current look-ahead window $\tau_{now}+\Delta\tau$.

The decision algorithm executes the following sequence of steps:

Step 1. *Look-ahead precondition check.* The look-ahead algorithm preliminary assigns resources on the ground of expected completion time of the tasks in T_{IP} (see Fig. 2) by conveniently updating matrix F_r. The look-ahead algorithm is executed only when (1) the number of free resources is lower than a predefined threshold, i.e. $|P_{ID}|<\delta_{ID}$, and (2) the number of unassigned tasks is greater than a threshold, i.e. $|T_{UA}|>\delta_{UA}$. This condition prevents the execution of look-ahead when the number of idle resources is sufficiently large with respect to the number of unassigned tasks. Essentially, when the conditions for look-ahead hold true, the assignment algorithm considers the resources in P_{EJ} as *idle* and *preassigns* them to tasks in T_{UA} according to the rules described in the following.

Step 2. *Task assignment precondition check.* At each sample time, the task assignment algorithm is executed only if the overall number of resources in P_{ID} (or $P_{ID}\cup P_{EJ}$ if look-

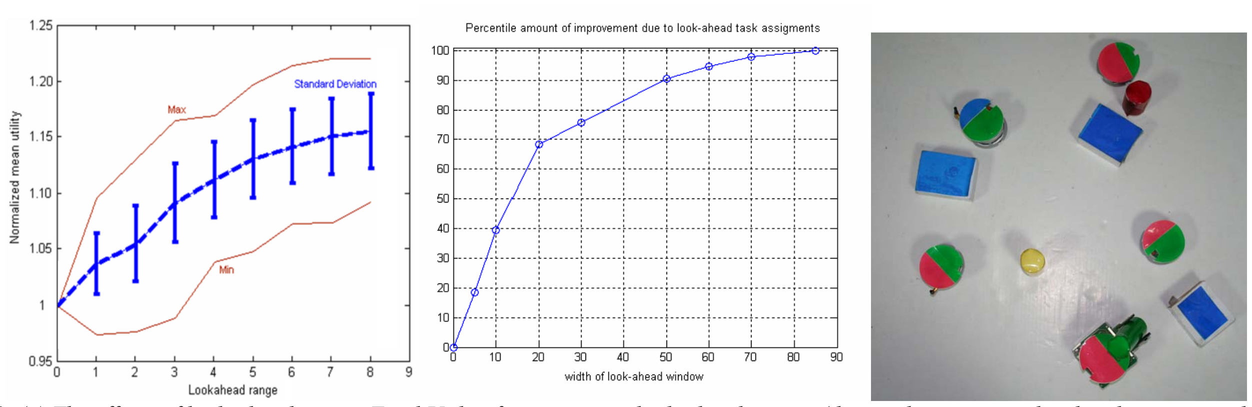

Fig. 2. (a) The effects of look-ahead range: Total Utility for increasing look-ahead ranges (the results are normalized with respect to the TU obtained without look-ahead). (b) Effect of increased width of look-ahead window (in time units). Simulations suggest to use high values for both parameters.(c) Top view (from the video sensor) of the experimental arena.

ahead is active) is above a predefined threshold δ_{PC} (significantly smaller than δ_{ID}). Otherwise, if $|P_{ID}|+|P_{EJ}| < \delta_{PC}$ the assignment is postponed: resources in P_{ID} (if any) are kept idle for few time samples in order to obtain a larger pool of alternatives for the assignment algorithm.

Step 3. *Initial assignment.* Each task is assigned to the resource in P_{ID} (or $P_{ID} \cup P_{EJ}$ if look-ahead is active) with the maximum utility, disregarding whether the resource has been already assigned to other tasks. If no assignment is possible, the task remains unassigned at the current sampling time.

Step 4. *Local Conflict Removal.* Tasks in conflict (i.e., assigned to the same resource) are sorted by flexibility (number of alternate resources that can perform the task). For each task in the sorted list, the algorithm checks if it can be redirected to a resource in P_{ID} at the cost of a decreased utility. Moreover, similarly to (Gage and Murphy, 2004), several heuristic procedures are applied to further reduce the amount of conflicts in the assignment. At the M-DEC level, conflict resolution strategies are implemented by suitably updating vector u_d and matrix F_{ud}.

Step 5. *Global conflict removal*. This step could be seen as an iterative refinement of the step 4. In this case, instead of redirecting tasks in conflict, the procedure tries to free some resource by redirecting tasks that are currently not in conflict, with the aim of making the freed resource available for tasks whose conflicts could not be removed in the previous step. Given the inherently combinatorial nature of this procedure, this step may become time-consuming even for relatively small numbers of tasks and sensors. For this reason, its execution is limited by a timeout (smaller than sampling time of the M-DEC). At the end of step 5, if conflicts remain, the M-DEC supervisory controller will command the start of the task with the highest utility, while all the other ones will be reconsidered at the next decision time.

5. SIMULATIONS

This section provides a summary of numerical simulations, directly obtained with the M-DEC implementation in Matlab platform, which have been performed to test and tune the free parameters of the task assignment algorithm, quantify the contribution of the look-ahead strategy and show the suitability of the M-DEC to be integrated with on-line dispatching strategies. The simulations consider a set of 294 randomly generated scenarios with 20 resources, and up to 400-500 tasks for each scenario. Each mission has a variable number of tasks which have randomly generated precedence constraints. Task characteristics and utility matrices (such as the one given as example in Table I) are also generated randomly. The scenarios describe problems with a variable degree of complexity, ranging from easy cases in which many homogeneous robots are available to perform few hardly overlapping missions, to more complex problems with many overlapping missions and heterogeneous robots).

TABLE I
EXAMPLE OF UTILITY TABLE.

	P1	P2	P3	P4	P5
L1	0.33	0	0	0.5	1
L2	0	0.67	0	1	0
L3	0	0	1	0	0
L4	0.8	0	0.9	0.33	0
L5	0	1	0	0	0.2
L6	1	0	0	0.67	0

Simulations are replicated altering one of the free parameters (thresholds δ_{ID}, δ_{UA} etc., look ahead window $\Delta\tau$). Results suggest that parameters δ_{ID} and $\Delta\tau$ are particularly critical for the effectiveness of the proposed task assignment algorithm. The sensitivity analysis with respect to δ_{ID} is summarized in Fig.2a. The mean utility (normalized with respect to the utility obtained without look-ahead) is increasing with the value of δ_{ID} (named *look-ahead range*). As a further example, Table II shows the data for a single simulation. It can be noticed that the increase of look-ahead range improves the overall utility at the cost of a slightly extended total makespan (time needed to complete all the missions in the scenario). Fig.2b summarizes the sensitivity of the results with respect to the look-ahead window $\Delta\tau$. Figures suggest that high values for both parameters permit to achieve the best combination of advantages related to look-ahead inspections. Similar analyses lead to conclude that good results are generally obtained when δ_{UA} is one to two times larger than δ_{ID}. (smaller values of δ_{UA} tend to leave too many resource unassigned; while larger values tend to increase resources' idle time).

TABLE II
EFFECTS OF INCREASING WIDTH OF LOOK-AHEAD RANGE

Look-ahead order	Makespan	Total Utility	Global Repair Intervention
0	1243	195.69	4
1	1243	195.69	4
2	1207	221.79	3
3	1213	247.14	5
4	1289	240.81	9
5	1247	255.66	9
6	1270	263.76	18
7	1273	263.00	9
8	1277	273.29	8
9	1330	272.73	10

6. EXPERIMENTS ON A MSN PROTOTYPE

The M-DEC control system has been implemented on an experimental laboratory-scale prototype of MSN, developed at the Laboratory of Robotics, DEE, Polytechnic of Bari. The MSN is composed of five mobile robots (K-Team Khepera robots) equipped with infrared sensors and encoders, and a set of transportable fictitious sensors that can be moved or removed by one mobile robot equipped with a small gripper with a contact sensor (Fig.2c). All the sensors are controlled by the M-DEC implemented in Matlab, and both sensory and control signals are transmitted by either radio-communication for battery-operated Kheperas or serial wires. The MSN has several tasks, which include "reach position", "seek for objects", "place sensor", "push obstacle", "map obstacle shape", "remove obstacle", etc. Battery-operated Kheperas are wireless: they have higher utilities for some tasks (e.g. they can reach any point in the arena), and lower ones for power-consuming tasks as "pushing obstacles". On the contrary, wired Kheperas have a navigation range limited by the wire, but can perform push or removal tasks with higher utilities. The approximated position of each sensor is obtained using a videocamera mounted on the top of the 1m×1m sensor arena (see Fig.2c). The M-DEC includes a textual interface for mission assignment, and a number of monitoring interfaces, which include the top view monitor.

The experiments performed on the prototype confirm the effectiveness of the integrated scheduling and control platform and its experimental feasibility. The M-DEC MSN completes autonomously all the missions issued through the textual interface, handling conflicts, synchronization of activities and feedback control in a unified modelling and control platform. Task assignments are generally performed with the highest utility and the control system tolerates very well the typical disturbances (noisy sensory data, delays due to image processing or communication through serial interfaces) of real-world devices.

7. CONCLUSIONS

This paper has presented a discrete-event coordination scheme for mobile heterogeneous sensor networks in the execution of multiple missions. The proposed control architecture integrates, in the same matrix-based formalism, a discrete event model, a dispatcher and a supervisory controller. In particular we have presented an effective dispatching algorithm that takes into account the estimated duration of a task (when available) to improve the overall utility of the assignment. As main result, this paper shows that the matrix formalism allows to (1) easily combine in a single framework task planning, dynamic resource assignment with look-ahead, and shared resource conflict resolution with utility-based method and to (2) easily optimize the controller parameters to cope with different scenarios, robot functionalities, MSN size. Also the use of a unified decision and control environment allows the designer to (1) model, (2) simulate, and (3) experimentally implement the closed loop system in a straightforward way.

The M-DEC proposed in this paper is fully centralized (it is run on a single PC). Following current trends in sensors networks, the research activities in progress are focused on the distribution of the M-DEC controller across a set of independent processors to achieve a multi-agent sensory platform.

8. ACNOWLEDGEMENT

This research work was partially funded by *Regione Puglia* under *Progetto Strategico CIP_PS121 "Telecommunication Facilities And Wireless Sensor Networks In Emergency Management".*

REFERENCES

Burgard, W., Moors, M., Stachniss, C., and Schneider, F.E., Coordinated multi-robot exploration; *IEEE Trans. on Robotics*, Vol. 21, N.3, June 2005.

Dharne, A.G.; Jayasuriya, S.; "A new protocol for the development and maintenance of autonomous mobile sensor networks", *Proc. of 2005 IEEE American Control Conf.*, 3494 – 3499, June 2005.

Farinelli, A., Iocchi, L., Nardi, D., Ziparo, V.A., "Task Assignment with Dynamic Perception and Constrained Tasks in a Multi-Robot System", ICRA 2005, *Proceedings of the 2005 IEEE Int. Conf. on Robotics and Automation*, 1523 - 1528 , April 2005.

A. Gage and R. Roberson Murphy, Sensor Scheduling in Mobile Robots Using Incomplete Information via Min-Conflict With Happiness *IEEE Trans. Systems, Man, And Cybernetics—Part B: Cybernetics*, Vol. 34, No. 1, February 2004

Gerkey B., Mataric M., "A formal analysis and taxonomy of task allocation in multi-robot systems", *The Int. Journal of Robotics Research*, Vol. 23, no. 9, pp. 939-954, September 2004.

V. Giordano, P. Ballal, F. Lewis, B.Turchiano, and J. B. Zhang, "Supervisory Control of Mobile Sensor Networks: Math Formulation, Simulation, and Implementation", *IEEE Trans. Systems, Man, And Cybernetics—Part B: Cybernetics*, Vol. 36, No. 4, August 2006.

J.-H. Lee, and H. Hashimoto, "Controlling Mobile Robots in Distributed Intelligent Sensor Network", *IEEE Trans. Industrial Electronics*, Vol. 50, No. 5, October 2003.

Mireles J., Lewis F., "Intelligent material handling: development and implementation of a matrix-based discrete event controller", *IEEE Trans. Industrial Electronics*, Vol. 48, Issue 6 , pp. 1087–1097, December 2001.

H. Ren, M.Q.H. Meng, X. Chen; "Investigating Network Optimization Approaches in Wireless Sensor Networks", *Proc. of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2015 – 2021; Oct. 2006.

D. Tacconi and F. Lewis, "A new matrix model for discrete event systems: Application to simulation," *IEEE Control Syst. Mag*., vol. 17, no. 5, pp. 62–71, Oct. 1997.

Tsalatsanis, A., Yalcin, A., Valavanis, K. P., "Automata-based Supervisory Controller for a Mobile Robot Team", *LARS '06. IEEE 3rd Latin American Robotics Symposium*, 53 - 59 , Oct. 2006.