A Multiobjective Dynamic Nonlinear Robot Assignment Problem

Sampo Ruuth, Helsinki University of Technology, Finland
István Maros, Imperial College, UK and University of Pannonia, Hungary
Kimmo Nieminen, Helsinki University of Technology, Finland

Imperial College, Department of Computing Technical Report 2007/3
ISSN 1469–4174

June 2007

Abstract

Robots will be used under rapidly changing and highly dangerous circumstances such as rescue operations in a radioactive environment or a fire as well as military operations. The robots are sent to several targets in order to carry out various tasks.

The robots we are considering here are able to send and receive messages to and from each other as well as solve nonlinear assignment problems. When the robot salvo is en-route to their targets several events may happen. A number of co-operative robots may get jammed as a consequence of disturbances. Some robots may already have reached their targets. Some robots may not be able to reach all targets. The system being investigated enables the surviving robots to work together in real time and change their pre-set tasks if necessary in order to maximize their effectiveness. In this paper we present a method which solves the reallocation problem using a piecewise linear network algorithm. Experimental results up to 493 targets and 500 robots show that the reallocation of the robots can be done in real time.

Key words: Co-operative robots, robot allocation, piecewise linear network, robot effectiveness, rescue operation.

1 Introduction

This paper is an updated version of an earlier publication [RU96] of the authors. It has been taken substantially further from both modeling and solution point of view.

Robots will be used under rapidly changing and highly dangerous circumstances such as rescue operations in a radioactive environment or fires as well as military operations. The robots are sent to several targets in order to carry out various tasks. It is possible to send more than one robot to a target. Depending on the communication capabilities, we distinguish the following types of robots.

1. *Co-operative robots*: can send and receive signals.
2. *Semi-cooperative robots*: can only send or receive signals but not both.
3. *Dumb robots*: can neither send nor receive signals.

When a group of robots is en-route to the destination of a group of targets several events may happen. It is inevitable that some robots will be damaged or destroyed (thus disabled). A number of cooperative robots may turn semi-cooperative or even dumb as a consequence of disturbed environment. Some robots may already have reached their targets and done their tasks. Some robots may no more be able to reach all targets. The system being investigated enables the surviving robots to work together in real time and change their pre-set targets if necessary in order to maximize their effectiveness.

Three communication concepts are considered which facilitate cooperation. They are:

- equal status robot concept,
- leader robot concept,
- remote control concept,

In equal status concept it is assumed that all cooperative robots are intelligent and of equal status. Information is reported to all other robots and all of them act according to identical algorithms (but maybe different data) loaded into the memory of the robots.

In leader robot concept a robot acts as a leader for the other robots. The leader acts upon data received from all the other robots and issues commands accordingly. All cooperative robots will be able to replace the leader robot in case the leader is disabled.

In remote control concept the robots communicate with the remote control center from where instructions are returned to them.

In the system described in this paper only cooperative or semi-cooperative robots are present. At the beginning all robots are supposed to be cooperative. During the mission robots cooperate when conditions change in order to determine whether a reallocation of the robots could improve the overall effectiveness of the group. In the current version of the model problem data at the cooperation time are generated randomly by the program after which a built-in network optimizer solves the reallocation problem.

The rest of this paper is organized in the following way. In section 2 we discuss the problem statement and model formulation, while in section 3 we present the solution algorithm and its pseudo code. The software implementation and computational results are described in section 4. The models and software developments undertaken in this study are discussed in section 5.

2 The Basic Model

The problem at each cooperation time for each concept and with changed external data described in Section 1 is to reallocate the robot group to the targets in real time so that the group effectiveness will be maximized and at the same time the number of changes from the robots' pre-set targets will be minimized.

The scoring scheme defining effectiveness is based on the definition of target score. Each target j $(j = 1, \ldots, n)$ is assigned a task success probability p_j and a weight $w_j \in \{0, 1, \ldots, 10\}$, which is a measure of the importance of the target. The probability that the task will be successfully carried out for that target depends on the number of robots y_j which have been assigned to the target in the following way:

$$1 - (1 - p_j)^{y_j}.$$

A *target score* is the product of its probability and weighting:

$$\text{target score: } \bar{g}_j(y_j) = w_j(1 - (1 - p_j)^{y_j}).$$

The robot *group effectiveness* is simply the sum of all individual target scores:

$$\text{group effectiveness } = \sum_{j=1}^{n} \bar{g}_j(y_j).$$

Let m denote the number of robots and $x_{ij},\ i = 1, ..., m$ and $j = 1, ..., n$, be the decision variables which equal 1 when robot i is assigned to target j and 0 otherwise. The problem can now be formulated as a nonlinear assignment type model. The nonlinearity, however, is present only in the objective function. Formally, the model can be described as follows, where i runs over robots $i = 1, \ldots, m$ and j runs over targets $j = 1, \ldots, n$:

Parameters:

m	number of robots
n	number of targets
w_j	denotes weighting of target j
p_j	denotes task success probability of target j
a_{ij}	$\in \{0,1\}$ is an entry of the $m \times n$ adjacency matrix indicating which target each robot can reach
u_j	upper bound on variable y_j (to be defined), $u_j = \sum_{i=1}^{m} a_{ij},\ j = 1, \ldots, n$

Decision variables:

$$\begin{aligned} x_{ij} &= \begin{cases} 1, & \text{when robot } i \text{ is directed to target } j, \\ 0, & \text{otherwise} \end{cases} \\ y_j &= \text{number of robots directed to target } j. \end{aligned}$$

Objective function:
The objective function is defined as the weighted sum of task success probabilities:

$$\max\ z = \sum_{j=1}^{n} \bar{g}_j(y_j) \tag{1}$$

Constraints:

$$\begin{array}{ll} y_j = \sum_{i=1}^{m} a_{ij} x_{ij}, & j = 1, \ldots, n \\ \sum_{j=1}^{n} x_{ij} = 1, & i = 1, \ldots, m \\ & \text{(every robot is assigned to exactly one target)} \\ x_{ij} \in \{0,1\}, & i = 1, \ldots, m \text{ and } j = 1, \ldots, n \\ y_j \le u_j & \text{(actually redundant but useful).} \end{array} \tag{2}$$

Looking at the model it is clear that the problem is not simply nonlinear but also integer.

3 A Solution Algorithm

3.1 Piecewise linear approximation

Assume that for each $i = 1, \ldots, m$ there exists at least one j such that $a_{ij} = 1$ (the problem has a feasible solution, i.e., initially each robot is assigned to a target). For realistic values of m and n, like $m = 30$ and $n = 7$, problem $(1) - (2)$ cannot be solved by complete enumeration in real time. Instead, because functions

$$\bar{g}_j(y_j) = w_j(1 - (1 - p_j)^{y_j})$$

are concave and monotone they can be approximated by piecewise linearization and the problem can represented as a separable network problem. Let us denote the piecewise linear approximation of $\bar{g}_j(y_j)$ by $g_j(y_j)$. Then we have

$$\begin{aligned}
\max \quad & z = \sum_{j=1}^{n} g_j(y_j) \\
\text{s.t.} \quad & g_j(y_j) = \sum_{k=1}^{u_j} s_{jk} y_{jk}, \quad j = 1, \ldots, n \\
& y_j = \sum_{k=1}^{u_j} y_{jk}, \qquad j = 1, \ldots, n \\
& 0 \le y_{jk} \le 1, \qquad j = 1, \ldots, n \text{ and } k = 1, \ldots, u_j.
\end{aligned} \tag{3}$$

where s_{jk} denotes the slope of the linear parts of $g_j(y_j)$ between break points $k-1$ and k, $k = 1, \ldots, u_j$ and y_{jk} is the fraction travelled between these two points. Function $g_j(y_j)$ is illustrated in Figure 1.

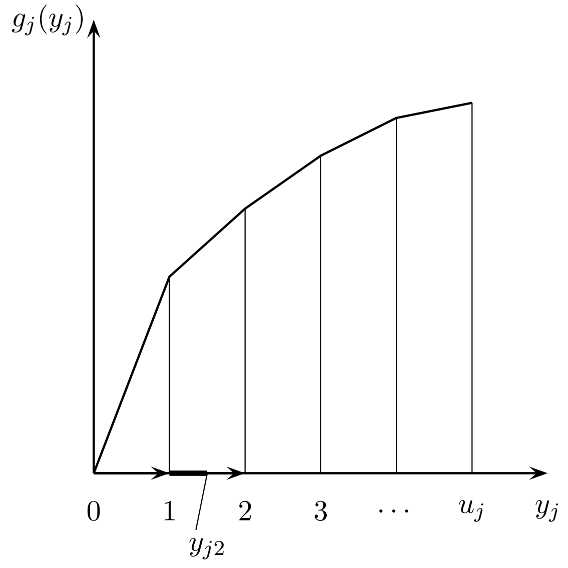
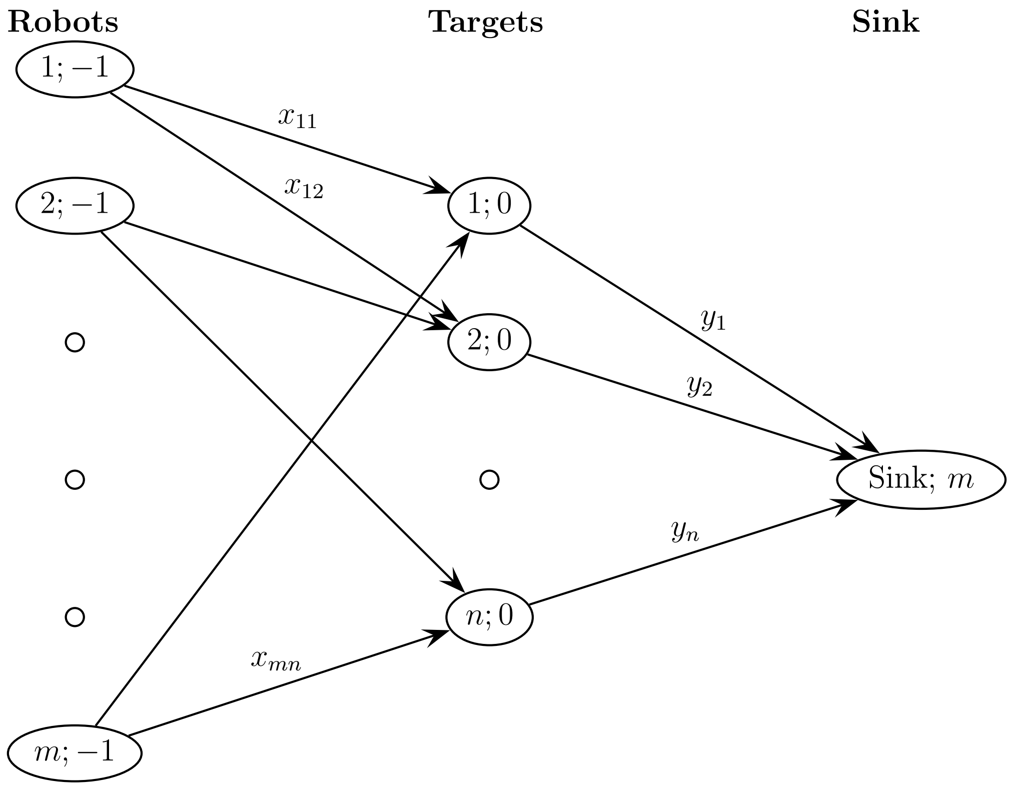


Figure 1: Piecewise linear approximation of function $g_j(y_j)$.

The model can be represented as a network which is illustrated in Figure 2. This network has a demand $= -1$ at the robot nodes, a demand $= 0$ at the target nodes and a demand = m at the sink node. The optimum solution of (3) is an integer solution because of the unimodular network structure of the constraints.



Node labels $(i; d)$: (node index; demand)

Figure 2: The model represented as a network .

3.2 Reoptimization

After the initial allocation the salvo of robots start the operation. As time passes by conditions where the robots are moving can change significantly. Before considering the reallocation of robots to targets (reoptimization) we have to take into account **for all concept types** whether a robot is *destroyed, jammed, damaged, which targets are inaccessible for it or it has already reached its target*. This is accomplished by modifying the matrix $a_{ij},\ i = 1, \ldots, m$ and $j = 1, \ldots, n$ according to the following rules.

(a) If a robot has been *destroyed* it is removed from the robot group. The number of robots is decreased by one.

(b) Assume a robot is *jammed*. If the concept is **remote control** the robot is fixed to its previously assigned target irrespective of whether the target is *possible* or not. This means that the target list of the robot (the ith row of matrix a_{ij}) is modified so that only the previously assigned target is included in it (the ith row of matrix a_{ij} contains only one 1). If the concept is **leader** the robot is not fixed to its previously assigned target because it can direct itself to all possible targets. If the concept is **equal status** no modifications are done to the target list of the robot because each robot determines itself to which target it will be directed.

(c) If a robot is *damaged* then in **all concept types** it is fixed to the lowest priority possible target or to a randomly selected target from the possible targets.

(d) If robot i cannot reach some targets any more (targets are not *possible* for this robot) then for **all concept types** these targets are removed from the target list of the robot (the corresponding column entries of the ith row of matrix a$_{ij}$ are set to 0).

(e) If a robot has *performed its task (reached the target)* then in all **concept types** this robot will be fixed to its previously set target.

Before the robots start their operation they are pre-assigned to some targets so that maximal a priori effectiveness is reached. At that time all targets are possible for each robot. Later on when significant events happen cooperation takes place, a_{ij} is modified according to the tests (a)–(e) above and the robots are reallocated to the targets if necessary.

Because of data communication uncertainty an optimum solution should also minimize the communication between two robots as well as between robots and the remote control center. In our model this is achieved by adding a linear term

$$\sum_i \sum_j c_{ij} x_{ij} \tag{4}$$

to the above defined piecewise linear objective function. Matrix c_{ij} is selected in the following way: $c_{ij} = x_{ij}$, for $i = 1, \ldots, m,\ j = 1, \ldots, n$, where x_{ij} is the optimal solution of the problem when it was solved previously. The values of decision variables y_j (the number of robots allocated to target j) most probably do not change because the additional term is small compared to the main term (piecewise linear term). It is also possible to multiply the main term by a sufficiently large number M to make sure that the values of variables y_j stay unchanged.

We now reformulate the problem. Again, i runs over robots $i = 1, \ldots, m$ and j runs over targets $j = 1, \ldots, n$.

Parameters:

m	number of robots
n	number of targets
w_j	denotes weighting of target j
p_j	denotes task success probability of target j
a_{ij}	$\in \{0,1\}$ is an entry of the $m \times n$ adjacency matrix indicating which target each robot can reach
c_{ij}	$\in \{0,1\}$ is an entry of an $m \times n$ matrix which equals 0-matrix when the allocation is made for the first time and equals the previous optimum solution matrix x_{ij} for later reallocations
s_{jk}	slopes of the piecewise linear approximations $g_j(y_j) = \sum_{k=1}^{u_j} s_{jk} y_{jk}$ of target score functions $\bar{g}_j(y_j) = w_j(1-(1-p_j)^{y_j})$
u_j	upper bound on variable y_j, $u_j = \sum_{i=1}^{m} a_{ij},\ j = 1,\ldots,n.$

Decision variables:

$$
\begin{aligned}
x_{ij} &= \begin{cases} 1, & \text{if robot } i \text{ is directed to target } j, \\ 0, & \text{otherwise} \end{cases} \\
y_j &= \text{number of robots directed to target } j. \\
y_{jk} &= \text{auxiliary variables associated with the piecewise} \\
&\quad\ \text{linear approximations of function } \bar{g}_j(y_j) = w_j(1-(1-p_j)^{y_j}).
\end{aligned}
$$

Objective function:

Now the objective function expresses the intention to maximize the weighted sum of task success probabilities and minimize the reallocations.

$$\max \sum_{j=1}^{n} g_j(y_j) + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{5}$$

Constraints:

$$\begin{array}{ll}
g_j(y_j) = \sum_{k=1}^{u_j} s_{jk} y_{jk}, & j = 1, \ldots, n \\
y_j = \sum_{k=1}^{u_j} y_{jk}, & j = 1, \ldots, n \\
y_j = \sum_{i=1}^{m} a_{ij} x_{ij}, & j = 1, \ldots, n \\
\sum_{j=1}^{n} x_{ij} = 1, & i = 1, \ldots, m \\
0 \le y_{jk} \le 1, & j = 1, \ldots, n \text{ and } k = 1, \ldots, u_j \\
x_{ij} \in \{0, 1\}, & i = 1, \ldots, m \text{ and } j = 1, \ldots, n
\end{array} \tag{6}$$

After the launch of the robots model parameters are exposed to significant changes. Therefore, reoptimization of the assignment problem will be necessary whenever a significant event occurs. For the repeated solution of problem (5)–(6) a dynamic algorithm can be used as shown in subsection 3.3.

3.3 Solution of the dynamic problem

For the solution of the allocation–reallocation problem (5)–(6) we propose the following dynamic algorithm:

Step 0. *Initialization*: Set $t = 0$ (time starts at zero). Set $a_{ij} = 1$ and $c_{ij} = 0$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Calculate the slopes s_{jk}, $j = 1, \ldots, n$, $k = 1, \ldots, u_j$. Solve problem (5)–(6) at the remote control center with equal status concept using a piecewise linear network solver. Denote the values in the optimal solution by x_{ij}^t. Assign the robots to their targets according to the optimal solution.

Step 1. *Waiting for a significant event*: Wait for a significant event until $t = T$ (initially a sufficiently long time). If significant event occurs go to **Step 2**. If $t = T$ go to **Step 4**.

Step 2. *Updating the dynamically changing model data*: Set $c_{ij} = x_{ij}^t$ for all combinations of i and j, and $t = t + 1$. Update the adjacency matrix a_{ij} according to rules (a)–(e) given in section 3.2.

Step 3. *Reallocation*: Solve problem (5)–(6). Change the target of the robots according to the optimum solution. Go to **Step 1**.

Step 4. *Termination*: Stop.

3.4 Solution of the initial allocation problem

Before the launch of the robot salvo the initial allocation problem ought to be solved, in other words we are looking for the solution for the problem of how many robots should be directed to a certain target in order to maximize the robot group effectiveness. A similar situation may also appear when the robot salvo is en-route to their targets. For some reason a robot may for instance loose completely it's pre-set allocation information.

To formulate the initial allocation problem set $a_{ij} = 1$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ in (1) – (2). We have

$$\begin{aligned}
&\max\ z = \sum_{j=1}^{n} w_j (1 - (1 - p_j)^{y_j}) \\
&\text{subject to} \\
&\sum_{j=1}^{n} y_j = m, \\
&y_j \geq 0 \text{ and integer},\ j = 1, \ldots, n.
\end{aligned} \tag{7}$$

The problem formulation does not specify to which target an individual robot will be directed. Thus the assignment of individual robots to a target can be done arbitrarily within the limits of the solution of (7).

Contrary to the general reassignment problem (5) – (6) the initial allocation problem can also be solved using *KKT (Karusch-Kuhn-Tucker) conditions* [LU03] in addition to piecewise linear optimization methods. For the KKT method we temporarily ignore the integer restrictions on variables y_j. Because the objective function is concave and the solution space is convex (defined by linear constraints) KKT conditions for the problem are both necessary and sufficient for optimality.

The initial allocation problem (7) is of form

$$\max\ z = f(y_1, \ldots, y_n)$$

subject to

$$\begin{aligned}
&g_0(y_1, \ldots, y_n) = 0 \\
&g_j(y_1, \ldots, y_n) \leq 0, \quad j = 1, \ldots, n
\end{aligned}$$

The KKT conditions for the problem can be written as

$$\frac{\partial}{\partial y_j} f(y_1, \ldots, y_n) - \lambda_0 \frac{\partial}{\partial y_j} g_0(y_1, \ldots, y_n) - \sum_{k=1}^{n} \lambda_k \frac{\partial}{\partial y_j} g_k(y_1, \ldots, y_n) = 0, \quad j = 1, \ldots, n$$

$$\begin{aligned}
&g_0(y_1, \ldots, y_n) = 0 \\
&g_j(y_1, \ldots, y_n) \leq 0, && j = 1, \ldots, n \\
&\lambda_j g_j(y_1, \ldots, y_n) = 0, && j = 1, \ldots, n \\
&\lambda_0 \text{ unrestricted in sign} \\
&\lambda_j \geq 0, && j = 1, \ldots, n
\end{aligned}$$

For problem (7) we have

$$\begin{aligned}
f(y_1,\dots,y_n) &= \sum_{k=1}^{n} w_k(1-(1-p_k)^{y_k}) \\
g_0(y_1,\dots,y_n) &= \sum_{k=1}^{n} y_k - m \\
g_k(y_1,\dots,y_n) &= -y_k,\ k=1,\dots,n
\end{aligned}$$

After proper substitution we obtain the KKT conditions for problem (7)

$$\begin{array}{ll}
-w_j(1-p_j)^{y_j}\log(1-p_j) - \lambda_0 + \lambda_j = 0, & j=1,\dots,n \\
\sum_{k=1}^{n} y_k = m & \\
\lambda_j(-y_j) = 0, & j=1,\dots,n \\
y_j \ge 0, & j=1,\dots,n \\
\lambda_0 \text{ unrestricted in sign} & \\
\lambda_j \ge 0, & j=1,\dots,n
\end{array} \tag{8}$$

In the pre-launch optimization it is reasonable to assume that $m \ge n$ and $y_j > 0$, $j = 1,\dots,n$ which means there are more robots than targets and every target gets at least one robot assigned to it. This implies that $\lambda_j = 0$, $j = 1,\dots,n$ and (8) becomes

$$\begin{array}{l}
-w_j(1-p_j)^{y_j}\log(1-p_j) - \lambda_0 = 0, j=1,\dots,n \\
\sum_{j=1}^{n} y_j = m \\
y_j > 0,\ j=1,\dots,n \\
\lambda_0 \text{ unrestricted in sign.}
\end{array} \tag{9}$$

For each $j = 1,\dots,n$ we solve y_j from (9) and get

$$\begin{aligned}
y_1 &= \frac{m - \sum_{k=2}^{n} \frac{1}{q_k}\log\frac{w_1 q_1}{w_k q_j}}{1 + \sum_{k=2}^{n}\frac{q_1}{q_k}} \\
y_j &= \frac{1}{q_j}\log\frac{w_1 q_1}{w_j q_j} + \frac{q_1}{q_j} y_1,
\end{aligned} \tag{10}$$

where

$$q_j = \log(1-p_j),\ j=1,\dots,n.$$

By rounding solution (10) to the nearest integer (in case $y_j < 1$ set $y_j = 1$) and making some simple manipulations an integer solution is found that is equal or very close to the optimal solution of the initial allocation problem (7).

4 Implementation and Computational Results

The initial allocation algorithm based on KKT method has been implemented in Borland C++ Builder 6 and the reoptimization algorithm in Dev-Pascal 1.9.2 both on a 2GHz Pentium PC using the object-oriented features of the programming languages. The code consists of about 2700 lines in all. It is divided into three sets of units:

- main set of units including the main program,
- algorithmic units which generate and update the model,
- network solver unit.

This division into sets of units allows us to make independent changes in individual units within a set of units.

In our tests random components are included in a unit where the adjacency matrix is updated. First the number of potential targets for each robot is generated randomly and after that the targets in question are randomly specified. Randomization was used to enable extensive computational study.

The number of robots in the models varies between 100 and 500 and the number of targets is between 100 and 493. The statistics of 25 randomly generated models are shown in Table 1. A typical salvo consists of 30 robots and 10 targets. There may be several successive launches of robots which all together can be viewed as a big salvo consisting of hundreds of co-operative robots and targets.

As we have seen the problem can be represented as a network (see Figure 2) with a nonlinear objective function. This function, however, is separable and concave and thus can be piecewise linearized with respect to each variable involved in it.

We have experimented with the following three schemes to solve the problems:

(a) Relax the integer restrictions in the initial allocation problem, solve the continuous problem by KKT method and round the continuous solution to the nearest integer solution such that all $y_j \geq 1$.

(b) Linearize each term of the objective function of both the initial allocation and reoptimization problems piecewisely and solve the linear network problems using a network solver.

(c) Solve the problem types directly using a nonlinear network solver which has a built-in piecewise linearizer.

Model size		Pre-launch allocation		Re-allocation	
Number of robots	Number of targets	Number of nodes	Number of arcs	Number of nodes	Number of arcs
100	100	201	11900	201	8189
100	200	301	23800	301	16925
100	300	401	35700	401	25066
100	400	501	47600	501	32677
100	500	601	59500	601	40660
200	100	301	21900	301	14335
200	200	401	43800	401	29922
200	300	501	65700	501	43392
200	400	601	87600	601	57291
200	500	701	109500	701	70809
300	100	401	31900	401	20479
300	200	501	63800	501	42683
300	300	601	95700	601	62259
300	400	701	127600	701	82060
300	500	801	159500	801	104677
400	100	501	41900	501	27239
400	200	601	83800	601	55252
400	300	701	125700	701	82723
400	400	801	167600	801	108037
400	500	901	209500	901	137240
500	100	601	51900	601	33695
500	200	701	103800	701	67852
500	300	801	155700	801	101176
500	400	901	207600	901	133391
500	493	994	255867	994	166074

Table 1: Robot allocation model statistics

In (a) we used the code that calculates the rounded KKT solution (10). In (b) we used network solver MINET presented in [MA87] and [MA88] and in (c) we used network solver PLNP which is based on a nonlinear network algorithm presented in [M97].

In Table 2 solution times for the models are shown for cases (b) and (c). The KKT solution times (case (a)) for all of the models were less than one millisecond and therefore they are not presented in the table. In column 1 of the table there are the solution times for the initial allocation problem, in column 3 and 2 the solution times for the reoptimization problem with and without the additional linear term (4) of the initial allocation solution in the objective function respectively. We observe that MINET is vastly superior to PLNP in all of the cases and the better the larger the model is. MINET is a general purpose minimal cost network optimizer. It has to solve a considerably larger problem [MA93] than PLNP which uses the compact form of the problem. On the basis of our experiences it can be said that the approach, KKT together with MINET, results in a formulation that can well be solved in fractions of a second. This means that only that one can be used for real time control of robots in the described environment.

Model size		1 Pre-launch allocation		2 Re-allocation **without** the additional term of the objective function		3 Re-allocation **with** the additional term of the objective function	
Number of robots	Number of targets	PLNP	MINET	PLNP	MINET	PLNP	MINET
100	100	0.06	0.03	0.04	0.02	0.03	0.00
100	200	0.15	0.07	0.10	0.02	0.05	0.02
100	300	0.27	0.11	0.18	0.03	0.09	0.04
100	400	0.40	0.14	0.28	0.07	0.19	0.05
100	500	0.55	0.21	0.34	0.07	0.22	0.06
200	100	0.21	0.10	0.14	0.05	0.07	0.03
200	200	0.48	0.19	0.31	0.08	0.20	0.05
200	300	0.82	0.28	0.55	0.12	0.28	0.09
200	400	1.32	0.39	0.81	0.13	0.50	0.11
200	500	1.64	0.52	1.13	0.17	0.56	0.14
300	100	0.47	0.19	0.31	0.07	0.16	0.06
300	200	1.00	0.36	0.70	0.12	0.45	0.11
300	300	1.64	0.59	1.14	0.22	0.64	0.17
300	400	2.33	0.75	1.67	0.25	1.08	0.20
300	500	3.12	0.90	2.33	0.32	1.10	0.30
400	100	0.82	0.31	0.57	0.11	0.30	0.10
400	200	1.70	0.65	1.25	0.19	0.61	0.18
400	300	3.10	0.78	2.02	0.31	1.04	0.27
400	400	3.95	1.36	2.86	0.38	1.71	0.33
400	500	5.03	1.61	4.09	0.50	1.98	0.41
500	100	1.27	0.39	0.92	0.15	0.43	0.13
500	200	2.60	0.81	2.13	0.26	0.98	0.22
500	300	4.08	1.16	3.14	0.42	1.52	0.35
500	400	5.92	1.72	4.51	0.56	2.45	0.47
500	493	7.39	1.98	5.74	0.65	3.64	0.55

Table 2: Robot allocation CPU times (sec)

5 Conclusions

We have presented a robot assignment model that has multiple applications. The problem has several key features. Robots act like communicating agents, the model is nonlinear and integer valued, furthermore, it has to be solved repeatedly if circumstances change during operation. An obvious requirement is the ability to solve the problem almost instantaneously as time is a decisive factor in the operation of these communicating robots.

We have proposed algorithms for solving the nonlinear integer model and have demonstrated that real-life-size problems can be solved practically in real time.

References

[LU03] Luenberger, D. G., *Linear and Nonlinear Programming*, Springer, 2003.

[MA87] Maros, I., *MINET a Fast Network LP Solver*, IIASA, WP–87–50, Laxenburg, Austria, June 1987.

[MA88] Maros, I., *MINET Fast Network LP Solver, Description and User's Guide for V2.00*, IIASA WP–88–6, Laxenburg, Austria, January, 1988.

[MA93] Maros, I., Performance Evaluation of the MINET Minimum Cost Netflow Solver, in Johnson, D. S. and McGeogh, C. C. (eds.) *Network Flows and Matching: DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Vol 12, 1993, pp. 199–217.

[M97] Marins, F. A. S., Senne, E. L. F., Darby-Dowman, K., Machado, A. F. and Perin, C., Algorithms for network piecewise-linear programs, *European Journal of Operational Research*, Vol 97, 1997, pp. 183–199.

[RU96] Ruuth, S., Maros, I., Lucas, C., Mitra, G., Solution of a nonlinear robot assignment problem, in Matson, E. (ed.) *Essays in honour of Bjorn Nygreen on his 50th birthday*, Norwegian University of Science and Technology, 1996, pp. 49–60.