Successful Search and Rescue in Simulated Disaster Areas

Alexander Kleiner, Michael Brenner, Tobias Bräuer, Christian Dornhege,
Moritz Göbelbecker, Mathias Luber, Johann Prediger,
Jörg Stückler, and Bernhard Nebel

Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
{kleiner, brenner, braeuer, dornhege, goebelbe, luber, prediger,
stueckle, nebel}@informatik.uni-freiburg.de

Abstract. RoboCupRescue Simulation is a large-scale multi-agent simulation of urban disasters where, in order to save lives and minimize damage, rescue teams must effectively cooperate despite sensing and communication limitations. This paper presents the comprehensive *search and rescue* approach of the *ResQ Freiburg* team, the winner in the RoboCupRescue Simulation league at RoboCup 2004.

Specific contributions include the predictions of travel costs and civilian lifetime, the efficient coordination of an *active* disaster space exploration, as well as an any-time rescue sequence optimization based on a genetic algorithm.

We compare the performances of our team and others in terms of their capability of extinguishing fires, freeing roads from debris, disaster space exploration, and civilian rescue. The evaluation is carried out with information extracted from simulation log files gathered during RoboCup 2004. Our results clearly explain the success of our team, and also confirm the scientific approaches proposed in this paper.

1 Introduction

The RoboCupRescue simulation league is part of the RoboCup competitions and aims at simulating large scale disasters and exploring new ways for the autonomous coordination of rescue teams [8]. These goals are socially highly significant and feature challenges unknown to other RoboCup leagues, like the coordination of heterogeneous teams with more than 30 agents, the exploration of a large scale environment in order to localize victims, as well as the scheduling of time critical rescue missions. Moreover, challenges similar to those found in other RoboCup leagues are inherent to the domain: The simulated environment is highly dynamic and only partially observable by a single agent. Agents have to plan and decide their actions asynchronously in real-time. Core problems in this league are *path planning*, *coordinated fire fighting* and coordinated *search and rescue* of victims.

This paper presents the comprehensive *search and rescue* approach of the *ResQ Freiburg* team, the winner in the RoboCupRescue Simulation league at RoboCup 2004. We show how learning and planning techniques can be used to provide predictive models that allow to act rationally despite the high dynamics of the simulation. Specific

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 323–334, 2006.

contributions include the prediction of the life-time of civilians based on *CART* [4] regression and *ADABoost* [6], travel cost prediction and its integration into target selection, the efficient coordination of an *active* disaster space exploration based on sensing, communication and reasoning, as well as an any-time rescue-sequence optimization based on a genetic algorithm. These techniques have in common that they provide ResQ agents with formal models for reasoning about the present and future state of the simulation despite its high dynamics. These models allow deliberative instead of purely reactive behavior, a capacity that we believe to be the reason for our team's success.

We have conducted an extensive comparison of the performance of our team with the performance of other teams in terms of the capability of extinguishing fires, freeing roads from debris, disaster space exploration, and civilian rescue. The evaluation is carried out with information extracted from simulation log files that were gathered during the RoboCup competition 2004. This evaluation gives much more information about a team's strengths and weaknesses than the standard scoring in the RoboCup Rescue Simulation league; yet it can be automated and therefore provides an instrument for precise analysis of teams. The results of our study clearly explain the success of our team, and also confirm the scientific approaches proposed in this paper.

The remainder of this paper is structured as follows. We present the active search and exploration approach in Section 2. The civilian rescue based on sequence optimization is described in Section 3. Path planning and travel cost prediction are covered in Section 4. Finally, an extensive evaluation and analysis of the 2004 RoboCupRescue competition is given in Section 5 and concluded in Section 6.

2 Exploration

In a partially observable environment like the RoboCupRescue simulation, exploration is the key means for agents to enlarge their knowledge. It is especially important to find injured civilians as quickly as possible without losing time by redundant exploration of the same area by several agents. Our agents achieve this ability by maintaining a *Knowledge Base* (KB) that keeps track of information collected on civilians during the search. Each agent maintains locally its own KB that is updated from senses, communication and reasoning. The KB allows them to efficiently focus and coordinate the search for civilians. It maintains the knowledge of an agent on the relation between the set of civilians C and the set of locations L. This is carried out by maintaining for each civilian $c \in C$ a set of locations L_c that contains all possible locations of the civilian. Furthermore, we maintain for each location $l \in L$ a set of civilians C_l that contains all civilians that are possibly situated at location l. Initially, $\forall c \in C, L_c = L$ and $\forall l \in L, C_l = C$.

The KB allows us the calculation of the expectation of the number of civilians situated at any location l. This is achieved by calculating the probability that civilian c is situated at location l, given the current state of the knowledge base:

$$P(loc(c) = l \mid KB_t) = \begin{cases} \frac{1}{|L_c|} & \text{if } l \in L_c \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Which yields the expectation on the number of civilians situated at location l:

$$E[|C_l|] = \sum_{i=0}^{|C|} P(loc(c_i) = l \mid KB_t) \tag{2}$$

Note that it follows from the above that initially the expectation for each location l is given by $E[|C_l|] = \frac{|C|}{|L|}$. That means that we expect civilians to be uniformly and independently distributed on the map. This is clearly not the case if buildings have a different size or different degree of destruction. As an improvement, one could incooperate this information as well. The KB is updated by either visual or auditory perception, communication of perception from other agents, and reasoning with respect to the agent's sensor model. The sensor model returns for any location l the set of locations V_l and A_l that are in visual (10m) or auditory (30m) range of l, respectively [11].

The KB is implemented as a $|C|x|L|$ boolean matrix, whereas C is the set of civilians and L the set of locations. Any entry $\langle c, l\rangle$ is set to *false* if a civilian c is definitely not at location l, and set to *true* otherwise (including the case of uncertainty). Initially, all entries are set to *true*. Based on the sensor model, one can perform either *positive* or *negative* update operations on the KB:

1. Positive updates:
   (a) **Civilian c seen at location l**
   We can reduce the set of possible locations for civilian c to l: $L_c := \{l\}$ and reduce [1] the set of possible civilians at location l to c: $C_l := \{c\}$;
   (b) **Civilian c heard at location l**
   We can remove civilian c from all civilian lists that are not in range of the sensor: $\forall l' : l' \notin A_l \Rightarrow C'_l := C'_l \setminus \{c\}$ and reduce the set of possible locations for civilian c to all locations that are in range of the sensor: $L_c := L_c \cap A_l$
2. Negative updates:
   (a) **Civilian c not seen at l**
   We can reduce the set of possible locations for civilian c by the set of locations within visual range: $L_c := L_c \setminus V_l$ and remove civilian c from all civilian lists for locations within visual range: $\forall l' : l' \in V_l \Rightarrow C'_l := C'_l \setminus \{c\}$
   (b) **Civilian c not heard at l**
   No safe conclusion possible

These update rules are very efficient due to the fact that the perception of the agents is free of noise. It is assumed that the agent is always able to see any civilian within the range of its sensors. Certainly this is not true in a real disaster situation. If, for example, victims are covered by rubble, they are even for humans hard to see. However, the proposed update rules can easily be enhanced towards probabilistic methods if there are probabilistic sensor models, which in turn have to be supported by the RoboCupRescue simulation system.

District exploration. District exploration is a multi-agent behavior for the coordinated search of buried civilians. The behavior guarantees that at any time each agent is assigned to a reachable and unexplored district on the map. In order to minimize the number of times an agent gets stuck in a blockade during exploration, districts have to consist of highly connected locations. The connectivity of two locations results from the number of alternative paths between them, the number of lanes and degree of blockage of each single road, and the degree of uncertainty on the state of the road. Due to the fact that blockades on the map and hence the map's connectivity is unknown to the agents

[1] Note if we see more than one civilian at the location, the set of possible civilians at l has to contain all of them.

in advance, the clustering has to be revised continuously. We used *agglomerative* [3] and *KD-tree* [2] based clustering in order to calculate a near optimal separation of the map into districts and to approximate the connectivity between them. These methods calculate from a given connectivity graph $G = \langle V, E \rangle$ of a city, where V represents the set of locations and E the set of connections between them, a hierarchical clustering. The hierarchical clustering, represented by a binary tree, provides at each level a partitioning of the city into n districts, reflecting the reachability of locations on the map (e.g. locations with a high connectivity are found within the same cluster). Based on this clustering, each team member is exclusively assigned to one reachable and unexplored cluster that represents a district on the map.

Active Exploration. Active exploration is an extension to the previously described district exploration task in that the search focuses on locations with high evidence on civilian whereabouts. This is carried out by exploiting the knowledge collected from senses, communication, and reasoning in the KB. Evidence from the KB is utilized by calculating an utility value $U(l)$ which is equal to the number of civilians expected to be found at observable locations O_l:

$$U\,(l) = \sum_{k \in O_l} E[|C_k|] \tag{3}$$

which yields, after inserting equation 2:

$$U\,(l) = \sum_{k \in O_l} \sum_{i=0}^{|C|} P(loc(c_i) = k \mid KB_t) \tag{4}$$

The overall sum of utilities over time can be maximized by the selection of targets with high utility as well as targets that are reachable within a short amount of time. Hence, from the set of locations L_D that are within the agent's district, a target location l_t is decided based on the trade-off between utility $U\,(l)$ and travel cost $T\,(l)$:

$$l_t = \underset{l \in L_D}{argmax}\, U\,(l) - \alpha * T\,(l) \tag{5}$$

whereas α is a constant regulating the trade-off between the estimated travel costs and the exploration utility and has to be determined experimentally. The estimated travel costs $T\,(l)$ are provided by a path planner that estimates costs based on a pre-calculated Dijkstra matrix (see Section 4).

Active surveillance. Furthermore, it is important for the rescue team to have up-to-date information on the injured civilians that have been found during the exploration task. The prediction module, described in Section 3, provides predictions of the civilian life time that are the more accurate the more up-to-date the information on *buriedness*, *damage* and *health* is. As we will describe in Section 3, the number of civilians that can be rescued depends on the efficiency of the rescue team, which in turn, depends on the accuracy of predictions. Hence, we extended the active exploration behavior in that it assigns agents to the surveillance of known civilian locations after the map has been explored sufficiently. The surveillance behavior is carried out by randomly sampling

interesting locations from the set of known civilian locations whereby locations with obsolete information are selected with high probability. In general, information on locations are considered as obsolete if they haven't been visited by agents for a long time.

The number of agents that are assigned to active search is limited to $\frac{|L|}{k}$, whereas L is the set of open locations and k a constant that has to be determined experimentally. All agents above the assignment limit are performing active surveillance. If $k = 1$ then there will be at least as many explorers as open locations. If $k < 1$ then the exploration speed will be increased, but in turn there might be obsolete information on the health state of known victims. If $k > 1$ then the quality of information on known victims will increase, but the search for new agents might take more time.

Team coordination. Besides the team coordination due to the assignment of districts, it is necessary to further coordinate the multi-agent search in order to prevent the multiple exploration of locations. This is carried out by communicating the information on found civilians as well as locations that have been visited. However, if agents select exploration targets from the same district (i.e. due to the overlap or the shortage of available districts), it might occur that they explore locations twice. We implemented two methods to locally reduce the probability of multiple target exploration. Firstly, agents select exploration targets from a probability distribution. Secondly, agents negotiate targets they plan to explore in the next cycle via the short range communication channel (*say* and *hear*). It turned out that the latter performs poorly if agents are able to move much longer distances in a cycle than they are able to observe, which is true for the current parameter setting of the RoboCupRescue kernel. The problem could be solved by performing the negotiation via the long-range communication. Unfortunately, this does no pay off since long-range communication is a limited resource. Hence, agents choose their exploration targets from a probability distribution that assigns to each target a probability that is proportional to the score following equation 5. Note that the local target selection strategy could further be improved by utilizing game-theoretic methods.

3 Civilian Rescue

Lifetime prediction. To achieve good results in the civilian rescue process, it is necessary to know a civilian's chance of survival. If there is a reliable prediction for the life time of a certain civilian, the scheduling of the rescue operation can be adapted accordingly. On the one hand, it is possible that a civilian does not need to be rescued at all because she will be alive at the end of the simulation. On the other hand, it is possible that a civilian would die within a short amount of time and therefore has to be rescued as soon as possible in order to survive.

For the ResQ Freiburg agents, machine learning techniques were used to gain a prediction of the civilian's life time and classification into survivors and victims. We created an *autorun tool* that starts the simulation and the agents simultaneously in order to collect data. The tool was used for several simulation runs on the Kobe, VC, and Foligno maps, from which a large amount of datasets were generated.

A data set consists of the values for *health* and *damage* of each civilian at each time step gained during the simulation. In order to reduce the noise in the data, simulations were carried out within 400 time steps, without rescue operations by the agents and

without fires on the map. The latter two conditions are necessary in order to prevent unexpected changes of the damage to a civilian due to its rescue, resulting in zero damage, or due to fires, resulting in unpredictable high damage. For the calculation of the life time, there has to be determined a time of death for each dataset. Hence, the simulation time was chosen to be 400 rounds, which seemed to be a good compromise between an ideal simulation time of ∞ and the standard simulation time of 300 rounds that would lead to a non-uniform distribution of the datasets.

Regression and classification was carried out with the *WEKA* [12] machine learning tool. We utilized the *C4.5* algorithm (decision trees) for the classification task. The regression of the simulation time is based on Adaptive Boosting (Ada Boost) [6]. Since the current implementation of the *WEKA* tool does only provide Ada Boost on classification, we had to extend this implementation for regression [5], which then has been applied with regression trees (CART) [4].

The regression trees have been evaluated on test data sets in order to learn the confidence of a prediction in dependency of the civilian's damage and the distance between the query time and the predicted time of death. Confidence values are necessary since the higher the difference between the observation and the civilian's actual time of death, the less accurate predictions are. The sequence optimization, described in Section 3, relies on the confidence values in order to minimize sequence fluctuations.

Genetic Sequence Optimization. If the time needed for rescuing civilians and the life time of civilians is predictable, one can estimate the overall number of survivors after executing a rescue sequence by a simulation. For each rescue sequence $S = \langle t_1, t_2, ..., t_n \rangle$ of n rescue targets, an utility $U(S)$ is calculated that is equal to the number of civilians that are expected to survive. Unfortunately, an exhaustive search over all $n!$ possible rescue sequences is intractable. A straightforward solution to the problem is, for example, to sort the list of targets by the time necessary to reach and rescue them and to subsequently rescue targets from the top of the list. However, as shown in Section 5, this might lead to unsatisfying solutions. Hence, we decided to utilize a Genetic Algorithm (GA) for the optimization of sequences and thus the subsequent improvement of existing solutions [7].

The time for rescuing civilians is approximated by a linear regression based on the buriedness of a civilian and the number of ambulance teams that are dispatched to the rescue. Travel costs between two targets are estimated by averaging over costs sampled during previous simulation runs. This is much more efficient than the calculation of exact travel cost involving, in the worst case, the calculation of the Floyd-Warshall matrix in $O(n^3)$.

The GA is initialized with heuristic solutions, for example, solutions that *greedily* prefer targets that can be rescued within a short time or urgent targets that have a short lifetime. The fitness function of solutions is set equal to the previously described utility $U(S)$. In order to guarantee that solutions in the genetic pool are at least as good as the heuristic solutions, the so called *elitism* mechanism, which forces the permanent existence of the best found solution in the pool, has been used. Furthermore, we utilized a simple one-point-crossover strategy, a uniform mutation probability of $p \approx 1/n$, and a population size of 10. Within each cycle, 500 populations of solutions are calculated by the ambulance station from which the best sequence is broadcasted to the ambulance teams that synchronously start to rescue the first civilian in the sequence.

One difficulty of the sequence optimization is given by the fact that information stored in the KB on civilians changes dynamically during each round and thus might cause fluctuations of the rescue sequence. This can be caused by two reasons: Firstly, civilians are discovered by active exploration, which is executed by other agents at the same time. Secondly, predictions vary due to information updates from active or passive surveillance. The latter effect can be weakened by updating the sequence with respect to the confidence of predictions. Updates of the information on civilians are ignored, if they are not statistically significant with respect to their confidence interval.

The effect of information updates due to exploration has to be controlled by deciding between rescue permanence and rescue latency, i.e. how frequently change the ambulances their targets and how fast can they react on emergency targets. Therefore we implemented additionally a reactive mechanism that recognizes emergency rescue targets that have to be rescued immediately. A target is defined as an emergency target if it would die if not being rescued within the next round. However, any other target is only taken as emergency target, if the current target would safely survive if postponing its rescue.

4 Path Planning

Every rescue agent must do path planning in order to reach its selected target position. ResQ Freiburg agents, however, use path planning already *during* target selection and thus can account for the time needed to reach a target when calculating its utility. Such an approach is only possible with a very efficient path planner that can be queried several hundred times in each cycle.

The efficiency of the ResQ path planner stems from the following realization (explained in more detail in [9]): many nodes on the road graph of a RoboCup Rescue map connect exactly two roads plus one or more houses. If none of these houses is the source or destination of the path planner query, the node can only be crossed, leaving no choices for the planner. Only nodes with more than two adjacent roads constitute real *crossings*. The road segments and simple nodes between the crossings can be joined in one *longroad*, which has no inner crossings. Longroads and crossings form a new, much smaller graph on which shortest path algorithms can be run much more quickly than on the larger original graph.

Since every node n from the original graph lies on a longroad, each path to or from n must include one of the two endpoint crossings of that longroad, c_n^1 and c_n^2. An optimal path from nodes s and e from the original graph therefore has length

$$\min_{i,j} \left(\overline{sc_s^i} + \overline{P(c_s^i, c_e^j)} + \overline{c_e^j e} \right) \text{ where } i, j \in \{1, 2\} \tag{6}$$

To solve this formula efficiently, the ResQ planner stores the direct routes from a location to its adjacent crossings. The optimal paths $P(c_s^i, c_e^j)$ between crossings are computed with Dijkstra's algorithm.

Adequacy of the path planner for the Rescue Domain is even more important than its efficiency. Most often agents want to know how long it will take to reach destinations. Therefore the cost functions used by the ResQ path planner have been designed not to return *path lengths* (although this is of course possible) but to predict the *time* it will take an agent to reach its destination. To compute this, the planner tries to consider

not only the length of a path, but also partial blockades, acceleration/deceleration at crossings, right of way (depending on from where a crossing is entered), and other agents' trajectories. While in the RCR system, these factors are accurately simulated, it is necessary for the ResQ Path Planner to use predictive functions in order to obtain the speed for several hundred queries per second.

We have provided several such prediction functions which, depending on the situation, use different aspects of an agent's knowledge about the world. For example, agents may sometimes want to choose only among roads that are known to be unblocked, but in other cases may ignore blockades completely in order to find out the minimal time to reach a target. Since the complex metrics used account for many of the specific influences mentioned above, we have been able to give a quite accurate prediction of travel durations in many cases. This prediction is then utilized by other components, e.g. the sequence optimizer for civilian rescue (cf. Section 3).

The simulation is cycle-based. Hence, finding paths with minimal lengths or even minimal duration is often not the wisest choice, since two paths differing only by a few meters or, respectively, a few seconds can often be considered as equivalent as long as they will take the same number of cycles to travel. This allows agents to build equivalence classes among paths and, consequently, targets. Several selection mechanisms allow to optimize other criteria when, for a set of targets, the expected number of cycles to reach them is equal. It is thus possible for an agent to select the most important target among the ones most easily reachable or, vice-versa, the closest among the most important targets.

5 Results

During the competition, teams are evaluated by an overall score that is calculated based on the state of civilian health and building destruction. However, since this score incooperates the total performance of all agent skills, such as exploration, extinguishing, and rescuing, it is impossible to assess single agent skills directly. In order to compare our agents with agents from other teams, the performance of typical agent skills are

Table 1. Percentage of clean roads

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	74,68	**82,22**	71,79	70,43	N/A	N/A	N/A	N/A
Final-Random	77,84	**86,51**	77,66	63,10	N/A	N/A	N/A	N/A
Final-Kobe	92,25	**93,74**	92,08	92,05	N/A	N/A	N/A	N/A
Final-Foligno	96,41	**97,72**	97,22	96,07	N/A	N/A	N/A	N/A
Semi-VC	67,93	**79,57**	68,86	57,90	67,22	57,85	53,27	80,53
Semi-Random	82,53	**87,44**	77,47	81,93	82,26	79,53	80,30	78,76
Semi-Kobe	92,40	93,65	92,71	92,51	92,62	92,56	93,55	**99,72**
Semi-Foligno	95,45	**97,08**	95,58	96,37	96,93	97,07	95,92	83,44
Round2-Kobe	92,52	93,52	91,46	92,46	92,78	93,45	92,25	**99,50**
Round2-Random	87,74	90,03	87,62	87,71	87,86	88,73	85,03	**99,97**
Round2-VC	91,34	91,62	90,74	89,87	91,40	90,92	N/A	**98,86**
Round1-Kobe	89,19	89,51	87,78	88,21	88,30	87,70	**91,12**	81,17
Round1-VC	91,90	92,13	91,74	91,84	N/A	91,81	91,54	**99,82**
Round1-Foligno	95,84	96,92	96,52	96,36	94,19	96,62	**97,63**	80,15
Number of wins	0	**7**	0	0	0	0	2	5
AVG %:	87,72	**90,83**	87,09	85,49	88,17	87,62	86,73	90,19
STD %:	8,25	**5,09**	8,59	11,25	8,93	11,59	13,63	9,96

Table 2. Percentage of saved buildings

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	47,21	54,13	**81,67**	43,19	N/A	N/A	N/A	N/A
Final-Random	24,04	**26,38**	15,03	12,35	N/A	N/A	N/A	N/A
Final-Kobe	38,24	**61,89**	38,38	13,51	N/A	N/A	N/A	N/A
Final-Foligno	**91,15**	62,77	60,92	34,56	N/A	N/A	N/A	N/A
Semi-VC	23,45	23,60	25,49	27,14	19,12	25,10	26,36	**27,22**
Semi-Random	23,18	**28,73**	18,09	19,55	22,82	21,45	17,09	18,91
Semi-Kobe	**96,49**	76,76	94,32	95,41	24,32	90,54	55,27	94,19
Semi-Foligno	36,22	**38,06**	32,72	37,79	31,89	28,48	26,82	23,23
Round2-Kobe	70,27	37,03	59,73	95,41	48,38	61,49	10,54	**95,54**
Round2-Random	99,04	60,91	54,68	99,16	63,55	97,60	80,70	**99,52**
Round2-VC	10,23	11,57	10,23	13,53	12,67	**71,99**	N/A	36,51
Round1-Kobe	99,46	98,92	**99,73**	**99,73**	99,05	98,78	67,16	91,89
Round1-VC	97,25	99,53	79,70	**99,76**	N/A	98,90	99,53	99,53
Round1-Foligno	**98,99**	**98,99**	36,13	45,99	32,53	54,29	43,59	29,86
Number of Wins:	3	**5**	2	2	0	1	0	3
AVG %:	61,09	55,66	50,49	52,65	39,37	**64,86**	47,45	61,64
STD %:	37,80	34,11	31,83	37,50	27,28	**31,63**	30,49	36,70

Table 3. Percentage of explored buildings

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	83,48	83,24	**87,02**	67,27	N/A	N/A	N/A	N/A
Final-Random	69,62	72,62	**78,13**	49,92	N/A	N/A	N/A	N/A
Final-Kobe	89,19	92,97	89,73	**94,19**	N/A	N/A	N/A	N/A
Final-Foligno	84,15	85,25	**86,73**	74,29	N/A	N/A	N/A	N/A
Semi-VC	69,39	72,86	**77,42**	45,08	52,01	52,87	47,92	59,72
Semi-Random	**78,91**	68,73	71,91	54,36	59,36	70,27	46,18	46,18
Semi-Kobe	85,41	96,22	92,97	95,54	66,62	97,30	**99,46**	91,89
Semi-Foligno	74,75	89,12	84,98	62,49	65,35	**92,53**	79,08	20,74
Round2-Kobe	87,16	90,68	95,00	91,76	80,54	94,19	**99,46**	92,43
Round2-Random	81,18	80,94	88,61	84,53	60,67	**94,24**	82,61	87,89
Round2-VC	83,40	70,18	84,58	40,44	67,74	87,88	N/A	**89,54**
Round1-Kobe	87,43	90,27	94,05	96,08	96,62	97,70	**97,84**	80,95
Round1-VC	85,37	90,48	95,28	94,26	N/A	97,72	**100,00**	91,35
Round1-Foligno	83,78	**90,05**	**90,05**	60,00	54,65	88,57	67,37	13,00
Number of Wins:	1	1	**4**	1	0	2	4	1
AVG %:	81,66	83,83	**86,89**	72,16	67,06	87,33	79,99	67,37
STD %:	5,82	9,98	**7,87**	22,21	13,87	14,59	21,84	30,80

emphasized by an evaluation of log files that were collected during the 2004 competition. The following tables provide results from all rounds of all teams that passed the preliminaries. All values refer to the last round, i.e. the percentage of clean roads at round 300. Bold numbers denote the best results that have been achieved during the respective round.

Table 1 shows the percentage of blockades that have been removed by the police agents. The results show that particularly the teams *Damas Rescue* and *The Black Sheep* most efficiently removed blockage from the roads. Table 2 shows the percentage of buildings that have been saved by the fire brigades. Obviously the team *Damas Rescue* saved most of the buildings, whereas *SBC* reached a robust behavior, shown by the good average value. The efficiency of exploration turned out to be one of the most important criteria for the team evaluation. The more locations of civilians are known, the more efficiently rescue operations can be scheduled. Table 3 shows the percentage of buildings

Table 4. Percentage of found civilians

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	97,22	94,44	**100,00**	81,94	N/A	N/A	N/A	N/A
Final-Random	**90,91**	85,71	81,82	70,13	N/A	N/A	N/A	N/A
Final-Kobe	**98,77**	97,53	95,06	**98,77**	N/A	N/A	N/A	N/A
Final-Foligno	**96,67**	**96,67**	**96,67**	72,22	N/A	N/A	N/A	N/A
Semi-VC	77,92	77,92	**85,71**	45,45	53,25	53,25	50,65	63,64
Semi-Random	**88,51**	73,56	72,41	63,22	67,82	80,46	52,87	55,17
Semi-Kobe	**100,00**	**100,00**	**100,00**	98,61	79,17	**100,00**	**100,00**	97,22
Semi-Foligno	90,12	95,06	86,42	81,48	83,95	**97,53**	85,19	30,86
Round2-Kobe	98,89	98,89	97,78	95,56	91,11	**100,00**	**100,00**	98,89
Round2-Random	**98,89**	95,56	**98,89**	81,11	70,00	96,67	85,56	94,44
Round2-VC	**92,22**	78,89	90,00	45,56	72,22	88,89	N/A	87,78
Round1-Kobe	94,29	**100,00**	**100,00**	98,57	**100,00**	**100,00**	94,29	78,57
Round1-VC	**100,00**	**100,00**	**100,00**	97,14	N/A	**100,00**	**100,00**	98,57
Round1-Foligno	**100,00**	97,14	94,29	77,14	74,29	92,86	77,14	14,29
Number of Wins:	**9**	4	7	1	1	5	3	0
AVG %:	**94,60**	92,24	92,79	79,06	76,87	90,97	82,85	71,94
STD %:	**7,17**	10,53	9,03	20,75	13,73	14,69	19,35	30,25

Table 5. Number of saved civilians

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	42	43	**52**	34	N/A	N/A	N/A	N/A
Final-Random	**32**	25	29	16	N/A	N/A	N/A	N/A
Final-Kobe	**46**	45	**46**	30	N/A	N/A	N/A	N/A
Final-Foligno	**66**	54	50	29	N/A	N/A	N/A	N/A
Semi-VC	**18**	15	17	12	11	12	12	14
Semi-Random	22	**26**	16	14	20	14	15	15
Semi-Kobe	**57**	47	54	52	20	39	34	44
Semi-Foligno	37	**46**	44	43	42	28	29	24
Round2-Kobe	**57**	37	43	50	43	35	28	43
Round2-Random	**52**	48	39	45	47	44	50	37
Round2-VC	31	33	32	24	37	**51**	N/A	34
Round1-Kobe	45	**51**	47	43	47	31	25	34
Round1-VC	**62**	**62**	55	57	N/A	51	54	44
Round1-Foligno	**53**	**53**	37	33	37	41	30	23
#Wins:	**9**	5	2	0	0	1	0	0
Σ TOTAL:	**620**	585	561	482	304	346	277	312
Σ SEMI+PREM	**434**	418	384	373	304	346	277	312

that were visited by agents[2]. The result shows that *Caspian* explored most of the buildings. However, the percentage of explored buildings does not necessarily correlate with the percentage of found civilians, as shown by table 4[3]. This is due to the fact that communication as well as reasoning might increase the efficiency of exploration. At the end, more civilians were found by *ResQ Freiburg* than *Caspian*, although the latter team explored more buildings. Important for efficient rescue operations is the point in time when civilian whereabouts are known. The earlier civilians are found, the better their rescue can be scheduled. Fig. 1 shows the number of civilians found during each cycle on the *RandomMap*. The results confirm the efficiency of *ResQ Freiburg's* exploration: At any time, the agents knew about more civilians than agents of any other team.

[2] Note: Full communication of visited locations and exploitation of a sensor model was assumed.

[3] Note: Civilians are considered as being found if one of the agents was within their visual range.

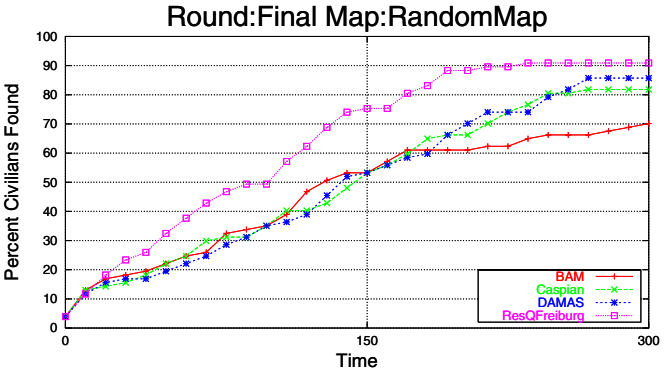

Fig. 1. Number of civilians found by exploration on a randomly generated map during the final

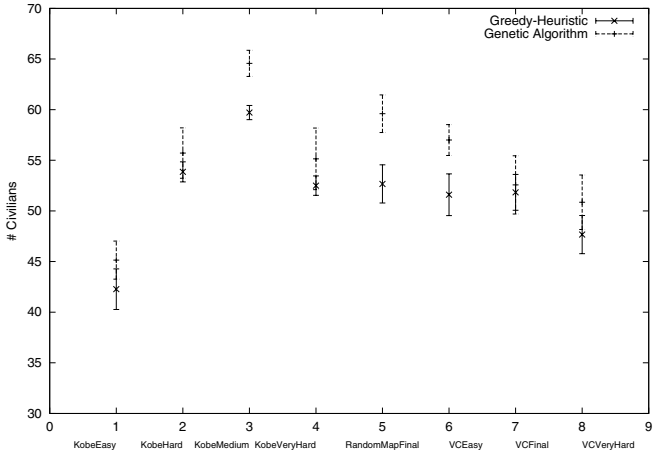

Fig. 2. Number of rescued civilians under two different strategies

Fig. 2 documents the difference between a greedy rescue target selection, i.e. preferring targets that can be rescued fast and selection based on an optimization by a genetic algorithm. It can be seen that an optimization of the rescue sequence clearly increases the number of rescued civilians. Finally table 5 shows the number of civilians saved by each team: *ResQ Freiburg* saved more than 620 civilians during all rounds, which are 35 more than the second best and 59 more than the third best in the competition.

6 Conclusion

The results presented explain the success of the *ResQ Freiburg* team during RoboCup 2004: While ResQ Freiburg's police agents (removal of blockades) and fire agents (extinguishing fires) performed comparably as good as agents from other teams (see table 2 and 1), the exploration and rescue sequence optimization abilities clearly outperformed the strategies of other teams (see table 4 and 5). Even during the competition's final on

the *RandomMap*, which decided by only 0.4 points of the total score the positioning between *Damas Rescue* and *ResQ Freiburg*, *ResQ Freiburg* was able to rescue seven civilians more than the second best.

In total, our results provide an interesting insight into the RoboCupRescue simulation competition: In addition to strategies for extinguishing fires and the removal of blockades, as they were favored by teams during the last years, exploration and sequence optimization are crucial subproblems in the RoboCupRescue simulation league. The proposed analysis provides a methodology for the further study of different strategies in this complex domain. The scoring metric for team evaluation shown in this paper has been integrated into the new 3D viewer of the RoboCupRescue simulation league, which we contributed for the next RoboCup competitions [10].

Currently, our team started to develop robots for the RoboCupRescue Real Robot league. We are confident that the methods proposed in this paper are also helpful in this context. Likewise as agents in the simulation, these robots have to find victims autonomously in an unknown terrain. Sensors, such as thermo cameras or CO_2 detectors, are used to make the search more efficient, in fact they are used to search for victims actively.

In addition to the proposed methods, various tools for agent world modelling and communication were developed by our team. These tools and also all algorithms discussed in this paper, are freely available for download from the official home page of RoboCupRescue simulation 2005 [1].

References

1. Homepage RoboCupRescue 2005. `http://kaspar.informatik.uni-freiburg.de/~rcr2005`, 2005.
2. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
3. H.H. Bock. *Autmatic Classification*. Vandenhoeck and Ruprecht, 1974.
4. L. Breiman, J.H. Friedman, R. A. Olshen, and C.J. Stone. *Classification and regression trees*. Wadsworth & Brooks, 1984.
5. Harris Drucker. Improving regressors using boosting techniques. In *Proc. 14th International Conference on Machine Learning*, pages 107–115. Morgan Kaufmann, 1997.
6. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
7. J. H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.
8. H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conf. on Man, Systems, and Cybernetics(SMC-99)*, 1999.
9. A. Kleiner, M. Brenner, T. Braeuer, C. Dornhege, M. Goebelbecker, M. Luber, J. Prediger, J. Stueckler, and B. Nebel. Resq freiburg: Team description and evaluation. Technical report, Institut für Informatik, Universität Freiburg, Germany, 2005.
10. A. Kleiner and M. Goebelbecker. Rescue3d: Making rescue simulation attractive to the public. `http://kaspar.informatik.uni-freiburg.de/~rescue3D/`, 2004.
11. T. Morimoto. *How to Develop a RoboCupRescue Agent*, 2002. http://ne.cs.uec.ac.jp/~morimoto/rescue/manual/.
12. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.