# The Effectiveness Index Intrinsic Reward for Coordinating Service Robots

Yinon Douchan and Gal A. Kaminka

**Abstract** Modern multi-robot service robotics applications often rely on coordination capabilities at multiple levels, from global (system-wide) task allocation and selection, to local (nearby) spatial coordination to avoid collisions. Often, the global methods are considered to be the heart of the multi-robot system, while local methods are tacked on to overcome intermittent, spatially-limited hindrances. We tackle this general assumption. Utilizing the *alphabet soup* simulator (simulating *order picking*, made famous by Kiva Systems), we experiment with a set of myopic, local methods for obstacle avoidance. We report on a series of experiments with a reinforcement-learning approach, using the *Effectiveness-Index* intrinsic reward, to allow robots to learn to select between methods to use when avoiding collisions. We show that allowing the learner to explore the space of parameterized methods results in significant improvements, even compared to the original methods provided by the simulator.

## 1 Introduction

Modern service robotics applications often rely today on multiple robots which coordinate tasks between them, and the user, in order to fulfill their design goals. Such multi-robot systems require coordination capabilities at multiple levels, from global (system-wide) task allocation and task selection, to local (nearby) spatial coordination to avoid collisions.

Yinon Douchan
School of Mechanical Engineering, Faculty of Engineering, Tel Aviv University, Israel, e-mail: yinondouchan@mail.tau.ac.il

Gal A. Kaminka
Computer Science Department and Gonda Brain Research Center, Bar Ilan University, Israel e-mail: galk@cs.biu.ac.il

Often, the global methods are considered to be the heart of the multi-robot system, as they are responsible for the scheduling and allocation of tasks to the robots. Local methods, on the other hand, are often added on, to overcome intermittent, spatially-limited hindrances to the tasks assigned to individual robots.

The Kiva Systems (Amazon Robotics) pick ordering and material handling system is a highly successful example of this design [13]. Here, hundreds of individual robots are sent around a warehouse to fetch and place back shelves containing products to be shipped to online customers. A centralized market allocation algorithm is used to optimize decisions as to which robot should move which shelf, and where. Each robot is then responsible to plan and execute a path from its current location to the target location, in order to maximize its own individual performance. Potential collisions between robots are resolved dynamically, ad-hoc[1].

This paper examines the role of such local, dynamic collision-avoidance methods within a multi-robot systems. We utilize the *alphabet soup* simulator [7], written by the founders of Kiva Systems to encourage research into their application domain. Here, as described in [13], a centralized optimization algorithm globally assigns tasks to robots. But planning paths and moving, to execute these tasks, is left to the robots.

We first experiment with a set of myopic, local methods for obstacle avoidance, which we demonstrate to work much worse than the methods provided with the simulator. Indeed, it turns out that it is quite difficult to get a local method to work well: the simulator's best method is a non-trivial stochastic combination of a myopic random direction selection, and predictive collision avoidance, a simple variant of the *dynamic window* algorithm [5].

Then, we report on a series of experiments with a reinforcement-learning approach using an *intrinsic* reward, to allow robots to learn which method to use when handling collisions. The reward function, called *effectiveness index* (EI), first proposed in [8], seeks to minimize resource usage when avoiding obstacles, and does not rely on communications or any information about the decisions or state of other robots. It is thus easily implementable in existing service robots, as it only relies on their own estimates of their own resource use.

While a first attempt at such learning generates mediocre results, we show that allowing the learner to explore the space of parameterized methods, results in significant improvements, even compared to the original methods provided by the simulator. The results demonstrate clearly that using the effectiveness-index reward, a reinforcement-learning technique can outperform manually-designed coordination methods, in a non-trivial real-world task. This, despite the simplicity of the learning mechanism itself. A second key lesson touches on the significant impact of local collision avoidance within a sophisticated multi-robot systems utilizing global coordination methods. The extreme differences in system performance when using different local methods—but all using the same centralized task allocation method—shows that good coordination cannot be carried out only at the global task alloca-

[1] This is actually not stated explicitly in [13], but is implied by the design, which explicitly leaves path-planning and motion-planning to each robot's individual controlling agent.

tion and team-plan level, but is instead an important factor at all levels of decision-making.

## 2 Related Work

Early work on local decision-making investigated reactive methods that proved useful in canonical multi-robot tasks, such as foraging, formations, or exploration. Balch and Arkin [1] describe a method, which we term *noise* in this paper, where once a robot is about to collide with another robot it moves backwards with some random directional noise. Vaughan et al. [12] describe *aggression*, a method in which when two robots collide with each other the robot with the highest aggression factor goes forward while the other backs away. Vaughan et al. describe three ways to determine the aggression factor for each of the two colliding robots: randomly, fixed, or based on each robot's free personal space behind them. We use the random version in the experiments reported below. Rosenfeld et al. [9] discuss the *repel* method, where by a collision is avoided by the robots moving backwards for a fixed amount of time.

Reactive methods are myopic, in the sense that they do not engage in, or rely on, any prediction of the future position or velocity of colliding robots. At a cost of more computation, a variety of sophisticated methods utilize predictions of others motions. *Dynamic window* [5] is one such method, which takes the motion constraints of the robot into account. It works by generating a search space for a navigation solution, and then selecting a heading and velocity within the space, that maximizes clearance from obstacles and other robots. The Reciprocal Velocity Obstacles (RVO) family of predictive methods takes into account not only the motions allowed for the robot, but also the responses of other robots to the these motions [11]. PassPMP [3] and related algorithms focus on safety in collision avoidance, at a cost of significant computation. In this paper, we utilize the *best-evade* and *original* methods supplied with the simulation (and seem to be variants of the dynamic window algorithm), and add the reactive methods discussed above.

Regardless of the method used—whether myopic or predictive—no method is always best. Both Rosenfeld et al. [9] and Rybski et al. [10] demonstrated that no local method is always better than others, but instead its performance depends on the density of robots. To address this, learning approaches have been utilized to try to dynamically adjust the collision-avoidance method to the density and other dynamic settings.

Rosenfeld et al. [9] measured a *Combined Coordination Cost* (CCC), the sum total of costs (time, energy, communications) that rise from a robot's attempt to coordinate away from collisions. The CCC was shown to be negatively correlated with performance in multi-robot foraging, and then used to guide dynamic selection of a reactive method for each robot. Each robot estimated its own CCC—relying on the estimation of the local density—and switched method if thresholds were passed. The thresholds were determined offline (before deployment), via hill-climbing. They do

not change with robot deaths, or with other dynamic changes to the task. Recent work by Godoy et al. [6] uses reinforcement learning to tackle the problem of multi-agent navigation. Their ALAN reward function controls the robot's velocity according to a weighted sum of two factors: a goal-oriented component factors the robot's distance to the goal; a second component considers how the robot's velocity vector affects others' velocities—using a variant of RVO to carry out this computation. A key difficulty with this reward function is that both of its components are *extrinsic*, relying on information external to the robot (e.g., the distance to the goal), which can be challenging to sense, and requires more sophisticated—and costly—capabilities.

To address this challenge, Erusalimchik et al. [8] proposed an *intrinsic* reward function, called the *effectiveness index* (EI). EI is a measure of the robot's own resource usage dealing with collisions. The robot only evaluates its own resources, and does not need any extrinsic information. However, EI has only been demonstrated to use in foraging, and its application in more structured settings is not straightforward (as we show) in particular where the navigation goals are given by a top-level decision process. We address this in this paper.

## 3 Reinforcement Learning Using Effectiveness Index

We briefly remind the reader of the EI reward function and its rationale. Please see [8] for details. The execution time of each robot's task can be divided into intervals where the robot is carrying out its task, uninterrupted, and these are interleaved with intervals where the robot is avoiding or otherwise actively handling an impending collision. This division of the task's time into interleaved execution and interruption intervals is characteristic of many service robotics tasks.

From the perspective of collision-handling there is a repeating pair of intervals: an active time interval, denoted $I_a$, where the robot is busy coordinating to avoid a collision, followed by a passive time interval $I_p$ where the robot performs its task. Given these, the EI for a given conflict, where a coordination method $\alpha$ was used, is

$$EI(\alpha) = \frac{I_a^\alpha}{I_a^\alpha + I_p^\alpha}. \tag{1}$$

This measures how costly a coordination method was (time spent $I_a$) against how effective it was (large $I_p$). The smaller EI is, the more effective the method. For example, assume it took for a robot 1 millisecond to coordinate but had a passive time of 1 millisecond afterwards EI will be 0.5. It took it a short time to coordinate but it also entered another conflict almost immediately. If it took a robot 5 seconds to coordinate but it had a passive time of 20 seconds then EI will be 0.2. It coordinated for quite a long time, but focused on the task afterwards for a much longer time and therefore, it is more effective.

Each robot independently uses the Q-learning algorithm with EI as a reward. We defined the action space to be the coordination methods. In particular, we use

a single-state version of Q-learning [4], $Q_t(\alpha) = Q_{t-1}(\alpha) + \rho(R_t(\alpha) - Q_{t-1}(\alpha))$, with $R_t(\alpha) = EI$. When a collision occurs, the robot calculates EI using the active and passive times before this collision and updates the Q-value of the method used in the previous collision. It then proceeds to making a selection as to the next method to be used.

## 4 Experiments in Learning Order Picking

We introduce the experiment test-bed and setup in Section 4.1. The following sections then systematically explore the use of learning in this environment.

### *4.1 Experiment Setup: Robotic Order Picking*

Order picking is the task of bringing products from various locations in a warehouse, to a centralized handling station where they can be packed together to fulfill an incoming order. It is a task carried out by people all around the world, and is considered to be particularly arduous: In some warehouses, pickers walk up to 15 miles a day in the warehouse.

In the mid 2000s, Kiva Systems began developing and selling robotic order picking systems, where robots would carry entire shelves and bring them to a human employee, who would be responsible for grasping the relevant products off the shelf, and packing them together. Kiva was sold in 2012 to Amazon, becoming *Amazon Robotics*, for a reported amount of $775 million.

#### 4.1.1 The Alphabet Soup Simulator

Alphabet soup is a simulator used to simulate a multi robot system in a warehouse [7]. It was developed by one of the Kiva Systems co-founders, in hope of encouraging research into algorithms that can be effective in this task. Multiple simulated robots deliver buckets containing letters to "word stations". Each word station contains words to be completed and robots must deliver letters for these stations in order to complete the demanded words. In order to place a letter in a word station, a robot must take the bucket, move it to the word station and take it back. There also exist letter stations in order to replenish a bucket's letter stash.

The simulator comes with a default set of settings (which we use, unless otherwise noted), including two local coordination methods: a *best-evade* method which makes a prediction as to a good obstacle-free direction, and a default *original* method, which is a stochastic combination of best-evade and random motions. The simulator also contains a task-allocation procedure, which makes allocations to the robots. We used it as is. A screen shot appears in Figure 1.

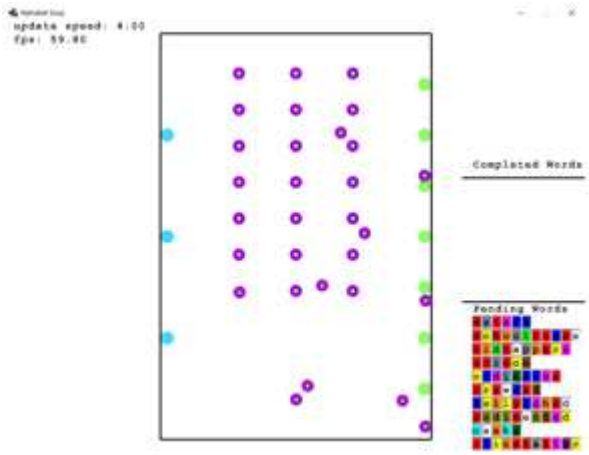


**Fig. 1** The Alphabet Soup simulator. Circles on the right side are the word stations, on the left side are letter stations, in the center are the buckets and the small lines are the robots.

#### 4.1.2 Coordination methods

Learning occurs over a set of base coordination methods, which can be applied to handle impending collisions. In addition to the best-evade and original methods, we implemented three additional coordination methods: *repel*, *noise*, and *aggression* (as discussed in Section 2). All methods, except *original* which cannot be adjusted, are time-based in the manner that they act for a limited time interval. For example, Repel with a parameter 200 milliseconds, given a conflict, will go back for 200 milliseconds.

#### 4.1.3 Initial Results Using EI

Despite its success in *foraging* [8], EI is not a magic bullet that works any time. Figure 2 shows the *baseline* results from applying EI to the five base methods, without any tweaking, and with arbitrary parameters for the different methods' timing set at 20ms. Each data point is the mean of 60 runs, each 10 minutes with measurement taking place only in the latter 5 minutes, to allow training to take place. The learning rate parameter $\rho$ was set to 0.5. Exploration rate was set at 0.1. In all the figures below, beginning with Figure 2, the X axis measures the number of robots within the fixed default working area. The Y axis measures the total number of letters (products) collected by all robots. Error bars mark a single standard deviation.

The results are shown to highlight that applying learning blindly does not work here, unlike in foraging [8]. The figure shows that the three reactive methods (*repel*, *noise*, *aggression*) are significantly and clearly inferior to the *original* and *best-evade* methods. Indeed, they do worse than random selection of coordination meth-

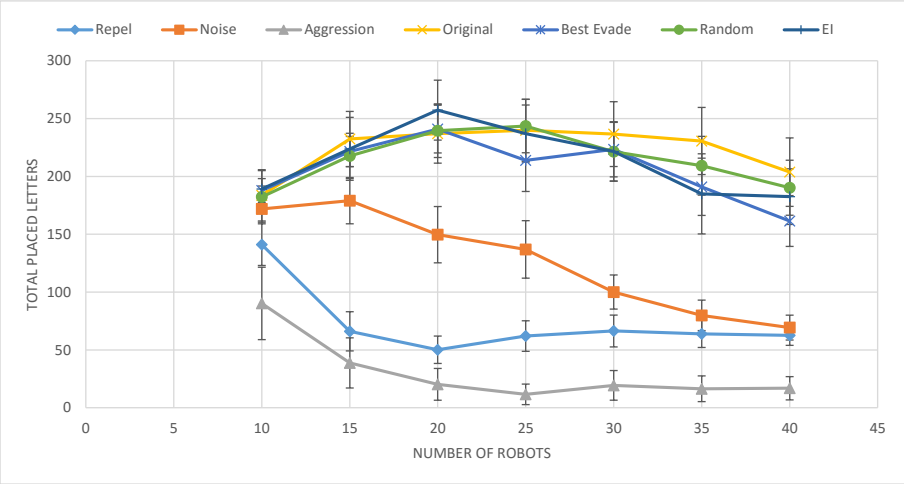


**Fig. 2** Initial results using *EI*.

ods (line marked *random*). EI-based learning improves on the base methods by combining them effectively (as each robot makes its own selections), but is only on par with *random* selection, the *Original* method and the *Best Evade* method.

## *4.2 Stateless EI Q-Learning with Parameters*

We have several directions to try to improve these results. One is to improve the base methods used in the learning, by allowing modification of the parameters used with each method. We demonstrate that a hierarchical learning of these parameters works much better than any manual tweaking. This is the approach we take in this section. In Section 4.3 we instead focus on enriching the state description at the basis of learning.

### 4.2.1 Choosing parameters according to area under curve

We first begin by attempting to choose the a single *best* parameter for each coordination method. To do this, we first conducted simulations for each of the method, for a variety of timing parameter values: 20, 100, 200, 500, 1000 and 2000. We then computed the area under curve for the resulting curves, with the rationale that the parameter with the largest area under curve would be the most reliable, across changes in density. The resulting parameters are: repel(200ms), noise(500ms), aggression(2000ms), and best evade(200ms).

Armed with this improved set of actions, we ran the EI learning. The results are shown in Figure 3. They show *repel* and *aggression* were the worst with at most

around 100 placed letters, only slightly improving on their earlier versions (different parameters). *Noise* improves clearly using the best parameter chosen for it. *Random* selection and the *EI*-based learned selection are now clearly distinguished: a random mix of these improved fixed-parameter methods does worse than earlier, while the learned selection improves on the *original* and *best evade* methods.

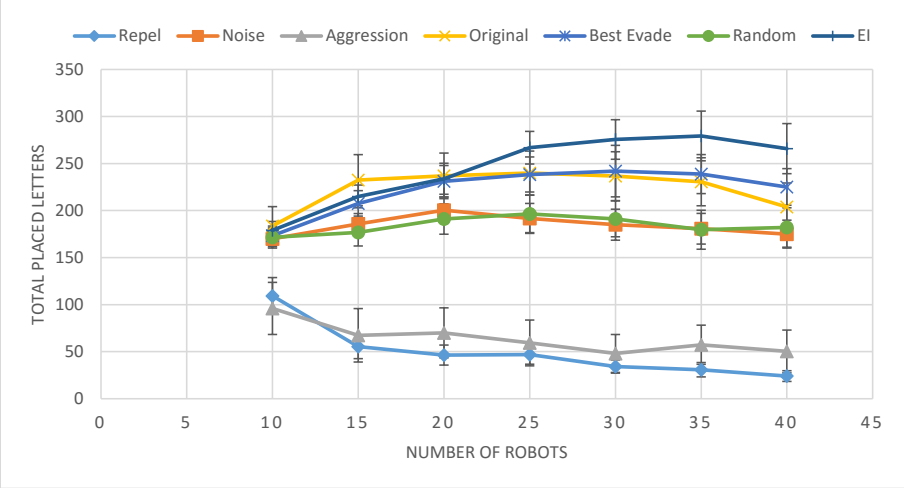


**Fig. 3** Applying stateless Q-learning [4] with EI reward, with reactive *best*-parameter methods.

Choosing parameters for coordination methods based on area under curve is a long and dull process, and offers no guarantees to produce good results, despite its relative success in this case. This is likely because the result of the process is a parameter that works well *on average*, and remains fixed through out the task. Indeed, we show that combinations of *worse* parameterized methods (in terms of their area-under-curve) can do *better*.

Figure 4 shows the results from using EI to learn when the underlying methods have been changed. Instead of using best-evade with a parameter of 200ms, we instead switch to using best-evade at 2000ms (which is worse, as the figure shows), and keeping all other methods as above. The combination of the other methods with best-evade 2000ms (worse by itself), does much better than with best-evade 200ms.

### 4.2.2 Stateless EI with Parameter Sweep

We now move away from fixed parameters, and use EI to not only choose the best coordination method, but also in order to adjust the parameters of the coordination methods. This was done by learning using a hierarchical version of EI and Q-learning. Q-learning with EI was used to select a general class coordination method. A second Q-learning with EI was used to learn the parameter that works best with

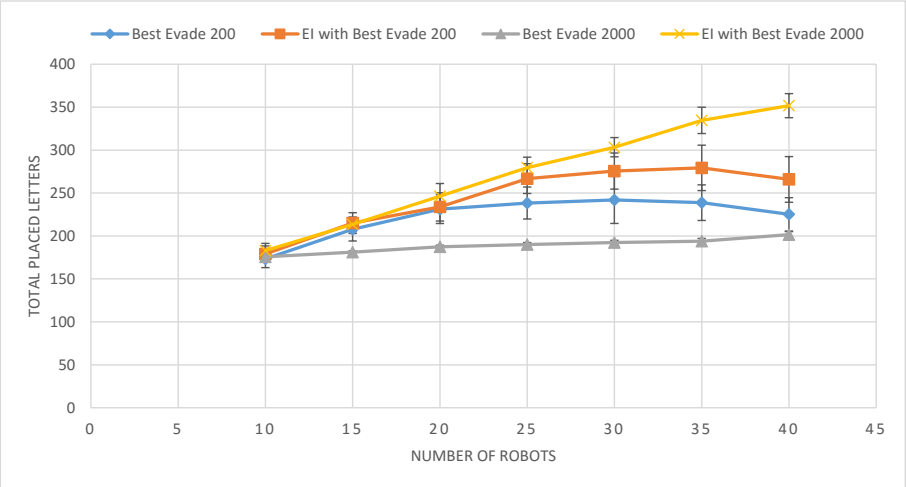


**Fig. 4** A comparison between Best Evade 200 and 2000 and stateless EI with best-evade 200 and best-evade 2000. X-Axis is the number of robots and Y-Axis is the total amount of letters placed.

this method. The two learning mechanisms run simultaneously, both using the same learning rate as before. The results, in Figure 5, show that learning also what parameter to select improves results significantly. To convince ourselves of the benefit of the learned parameters, we fixed the parameters for each method based on the results of the learning, and went back to trying Q-learning over the (small) set of methods, with the fixed parameters: Repel 700ms, Noise 540ms, Aggression 500ms and Best Evade 600ms. The results are shown in Figure 5, and match the simultaneous learning results.

## *4.3 Stateful (Multi-State) EI*

We had also experimented with a different approach to improving the performance of the team using reinforcement-learning and EI. In particular, in this section we report on experiments where the set of coordination methods was once again restricted to atomic, non-parameterize simple actions, while the Q-learning method was used to maintain multiple states. In this section we use the original Q-learning formula for multiple states $Q_t(s,\alpha) = Q_{t-1}(s,\alpha) + \rho(R_t(\alpha) + \gamma \cdot max_{a'} Q(s',a') - Q_{t-1}(s,\alpha))$ with $\gamma = 0.7$ and $\rho = 0.5$. Each data point is the mean of 10 runs, each 10 minutes with measurement taking place only in the latter 5 minutes, to allow training to take place.

We had explored two different ways of specifying state. The first state factor maintained position of the robot. This was done by a discretization of the work area into a grid (not necessarily regular), and keeping track of the grid cell the robot was

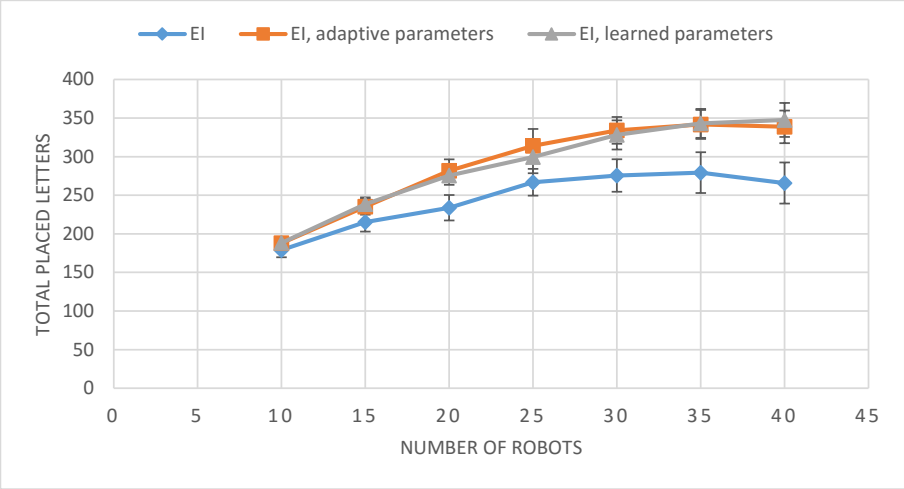


**Fig. 5** Stateless EI with fixed parameters learned from adaptation against EI with highest area under curve parameters. X-Axis is the number of robots and Y-Axis is the total amount of letters placed.

in. A second factor consisted of the heading of the robot. A final factor consisted of the task state of the robot (e.g., "taking a bucket to be processed" vs. "taking a bucket back to home position").

Using stateful (multi-state) Q-learning with these state spaces and the five aforementioned coordination methods as the action space yielded results that were, at best, like stateless EI. We therefore moved to defining new sets of coordination methods and tested stateful EI with them. We first used motion in absolute directions—West, East, North and South. Given a conflict the robot goes to this direction for a fixed amount of time. We arbitrarily determined the parameters of these methods to be 500 milliseconds. We ran the same simulations with the above set of coordination methods and compared group performance of random method selection, stateless EI and stateful EI with different state spaces.

Figure 6 shows that with absolute directions to resolve collisions, stateless EI improves performance significantly in comparison to random selection. However, stateful EI with direction as a state (40 Directions) does not improve results in relation to stateless EI. Using position on a $10 \times 4$ grid (40 positions) as the basis for the learning improved results significantly over the original method. However, it still is not as good as simply using stateless EI with parameter sweeping, as introduced in the previous section.

We have also experimented with trying relative motions, instead of absolute motions. Just like we can define absolute directions as coordination methods, we can define relative directions: Left, Right, Forward, or Back. Figure 7 shows the results with these relative motions. Clearly, they are quite disappointing: None of the state

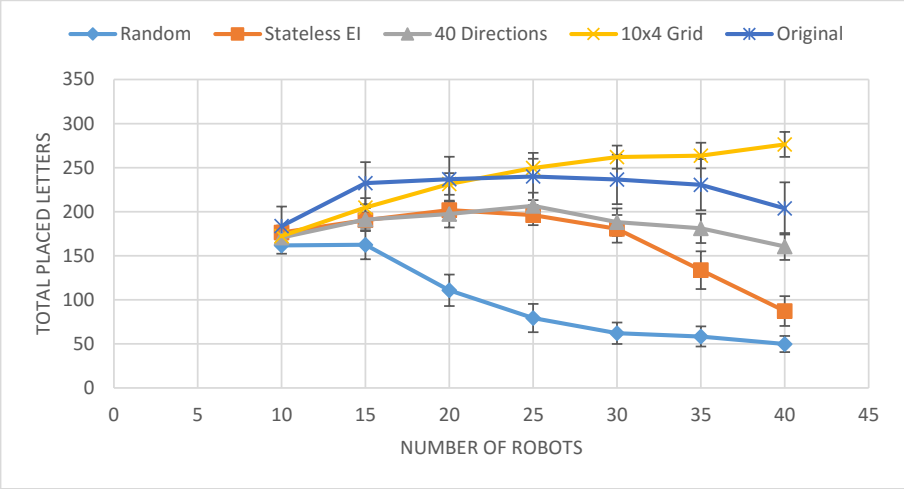


**Fig. 6** Learning over absolute actions: West, East North and South.

feature combinations tested lead to results that improve over the *original* method used in the simulation.

It is shown that with relative motions stateless EI does not improve performance over random selection. Stateful EI, using various combinations of state features, incrementally improves over stateless learning. First by incorporating the position on a $10 \times 4$ grid as the state, then using the heading (distinguishing 40 angles), and then finally in combination of position and heading. We have also attempted to incorporate the task state ("bucket to station", "bucket to storage", etc.) into the state, but this did not change the results.

What can be concluded from here is that the problem is not only choosing what state space, but what state space we should choose given an action space or what action space should we choose given a state space. This leaves an opening for a more formal investigation as to when a state-action space works better than another state-action space for a specific problem, given a learning method.

## *4.4 Comparison to ideal conditions*

In the previous sections we experimented with various learning techniques which improve performance to a certain extent. We would like to know how an ideal system—one with no collisions—will perform comparing to the best we achieved with learning. Therefore, following [9] we modified the simulator so that robots were able to go through other robots and measured the system's performance with this modification. We will call this modification *GoThru*. We ran simulations for

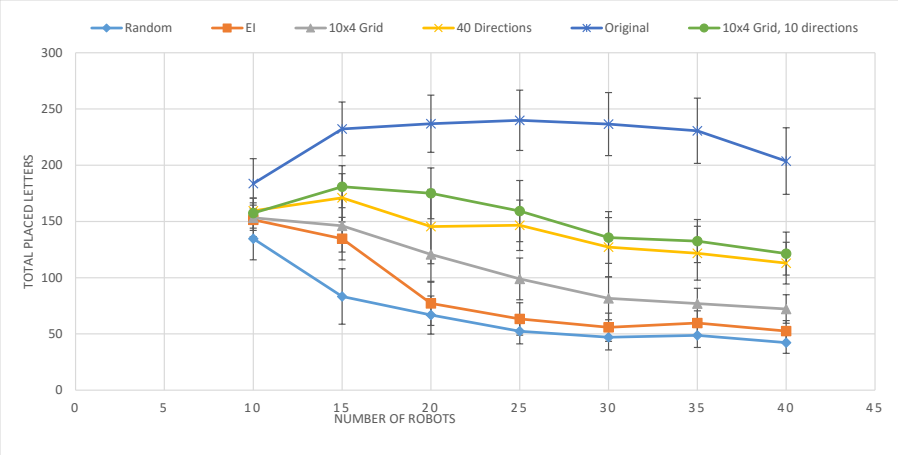


**Fig. 7** Relative directions Left, Right and Back. X-Axis is the number of robots and Y-Axis is the total number of letters placed.

GoThru in the same manner we did for EI. For GoThru averaging was done over 10 runs per sample and for EI over 60.

Figure 8 shows that GoThru performs at least almost 3 times better than EI and at most more than 5 times better than EI. This shows that not only there is a large room for improvement, but also shows the significance of local collision avoidance even in lower densities.

## 5 Summary and Future Work

This paper explores the use of an intrinsic reward function, called *effectiveness index*, applied to local multi-robot coordination, in particular in addressing impending collisions. The function is used with simple, plain Q-learning [4]. The results demonstrate clearly that with an appropriate set of base actions, a reinforcement-learning technique using the effectiveness index reward can outperform complex, manually-designed coordination methods, in non-trivial tasks for service robots. Indeed, the improved learning system utilizes simultaneous learning in two layers: the robots learned which method to select, while also learning which parameter to use for the selected method. We look forward to experimenting with additional advanced method for parameter learning (e.g., [2]).

A second key lesson touches on the significant impact of local collision avoidance within a sophisticated multi-robot systems utilizing global coordination methods. The extreme differences in system performance when using different local methods—but all using the same centralized task allocation method—shows that

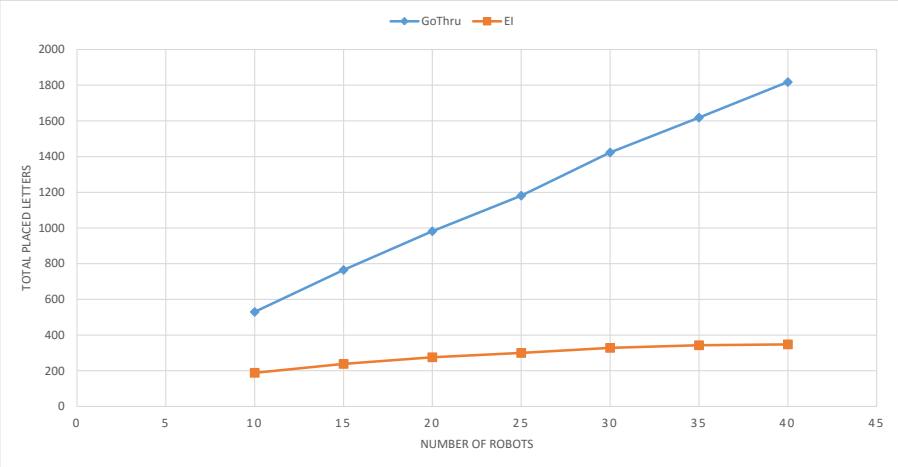


**Fig. 8** EI with learned parameters (best EI variant) compared to GoThru. X-Axis is the number of robots and Y-Axis is the total amount of letters placed.

good coordination cannot be carried out only at the global task allocation and team-plan level, but is instead an important factor at all levels of decision-making. In the future, we plan to apply this learning approach to adversarial settings, and to settings requiring even more complex, structured task execution.

**Acknowledgements** We gratefully acknowledge support by ISF grant #1511/12, and good advice from Avi Seifert. As always, thanks to K. Ushi.

## References

1. T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, December 1998.
2. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 12:281–305, 2012.
3. S. Bouraine, T. Fraichard, O. Azouaoui, and H. Salhi. Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.
4. C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, 1998.
5. D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, Mar 1997.
6. J. E. Godoy, I. Karamouzas, S. J. Guy, and M. Gini. Adaptive learning for multi-agent navigation. In *Proceedings of the Fourteenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, pages 1577–1585, 2015.

7. C. J. Hazard and P. R. Wurman. Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *In Proceedings of the 2006 AAAI Workshop on Auction Mechanisms for Robot Coordination*, pages 23–30, 2006.
8. G. A. Kaminka, D. Erusalimchik, and S. Kraus. Adaptive multi-robot coordination: A game-theoretic perspective. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-10)*, 2010.
9. A. Rosenfeld, G. A. Kaminka, S. Kraus, and O. Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6–7):633–655, 2008.
10. P. Rybski, A. Larson, M. Lindahl, and M. Gini. Performance evaluation of multiple robots in a search and retrieval task. In *Proceedings of the Workshop on Artificial Intelligence and Manufacturing*, pages 153–160, Albuquerque, NM, August 1998.
11. J. van den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. *Robotics Research*, pages 3–19, 2011.
12. R. Vaughan, K. Støy, G. Sukhatme, and M. Matarić. Go ahead, make my day: robot conflict resolution by aggressive competition. In *Proceedings of the 6th int. conf. on the Simulation of Adaptive Behavior*, Paris, France, 2000.
13. P. R. Wurman, R. DÁndrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, Spring, 2008.