Combining Re-Allocating and Re-Scheduling for Dynamic Multi-Robot Task Allocation

Yin Chen, Xinjun Mao, Fu Hou, Quizen Wang, Shuo Yang
College of Computer, National University of Defense Technology, Changsha, Hunan, China 410073
Email: chaanyean@163.com

Abstract—Multi-robot systems (MRS) working in open and dynamic environments are expected to deal with uncertain arrival of new tasks and environment changes, by repeatedly adapting the current task allocation and schedule, in order to maintain its performance (e.g., total utility, balance, etc.). This paper presents an adaptive approach to multi-robot task allocation (MRTA), which combines two adaptive measures corresponding to different levels of a MRS: (1) re-allocating at inter-robot level, for balancing task allocation, and improving total utility of the MRS, and (2) re-scheduling at intra-robot level, for maintaining each robot’s utility against the influence of both re-allocating and environment changes. Our approach is expected to have significantly higher adaptation power than both re-allocating only and re-scheduling only cases. An experiment is conducted to evaluate our approach’s capability of improving balance and total utility of the MRS, under different environment settings and different combinations of re-allocating and re-scheduling.

I. INTRODUCTION

Multi-robot systems (MRS) in dynamic environments often face tasks that are released at uncertain time and with changing locations. Examples of such systems include a robot guidance system in a shopping mall [7], a robot delivery system which delivers goods to the current location of the receiver [12], or an UAV system attacking mobile targets [1]. In these systems, a task point (an atomic task corresponding to a unique geographical location) can be related to a mobile entity (e.g., human or vehicle), of which the MRS has limited knowledge to predict its movement. In order to improve its performance, the MRS cannot rely on one-shot task allocation or scheduling (both conventionally assume a stationary environment) before task execution, since the advantage of the original allocation or scheduling results may soon be degraded when the dynamic entities have moved. Instead, the MRS has to dynamically adapt its task allocation and scheduling results in response to the influence of environment changes.

Fig. 1 analyses how environment changes influence the MRS’s performance. The source of the influence is task level (i.e., environment and task execution), which has following dynamic factors: (1) dynamic locations of task-related physical entities; (2) human requests released at uncertain times; (3) robot status, especially the location of each robot, that changes when the robot moves. The successive two levels are where these dynamic factors take effect on system performance: (1) intra-robot level, i.e., schedule of each robot, and (2) inter-robot level, i.e., MRS and its task allocation. At intra-robot level, the set of task points possessed by each robot will be changed by re-allocating and the release of new tasks; in addition, the movement of task-related physical entities will change the utility of each task point. For these reasons, re-scheduling is needed to compensate for negative effects of such changes, or exploit their positive effects, so as to improve the schedule’s utility. At inter-robot level, since new tasks may arrive, and existing task points are re-scheduled, the optimality of original task allocation (based on original schedule and task allocation) may easily get lost. Therefore it needs to repeat allocation for many times in order to tackle the influence of such changes. However, further re-allocating will also change the set of task points of each robot, forcing further re-scheduling as a response. Such iterative re-scheduling and re-allocating will persist until it finally reaches balance (i.e., unable to make significant improvement).

Corresponding to the inter- and intra-robot levels, we combine two adaptive measures: re-allocating and re-scheduling, in order to maintain MRS’s performance. In detail, our approach devises: (1) auction-based re-allocating, based on pairwise utility improvement, for balancing individual utilities and improving global utility, and (2) greedy re-scheduling, for improving individual utility; so as to yield adaptation power that outperforms both re-scheduling only and re-allocating only cases. We have conducted two groups of experiment (showing the instantaneous adjustment and accumulated effects respectively) in order to reveal the effectiveness and limitations of our approach in improving total utility and balance under different environment settings.

The rest of the paper is organised as follows. Section II
introduces related work. Section III gives a formal problem statement. Section IV presents our combinatory approach to dynamic task allocation. Section V presents and analyses the experiment results. Section VI concludes this paper.

II. RELATED WORK

Multi-robot task allocation (MRTA) has received wide attention in recent years [2][3]. Decentralised MRTA approaches, especially market-based approaches using auctions [4], have gained popularity due to their capability of overcoming the limitations of the centralised solutions, e.g., low responsiveness to changes, vulnerability to failures, high computation and communication cost, intractability for large-scale problems [1], etc. Since environment changes are common to many robot applications in open environments, dynamic MRTA is needed, in which task allocation is continuously adjusted [5]. Currently several researches have addressed MRTA with dynamic and/or moveable tasks. B. Heap et al [12] deal with dynamically inserted pickup and delivery tasks considering the tradeoff between solution quality and planning time. G. Li, Y. Tamura et al [8] deal with dynamic sequential task allocation and re-allocation based on body expansion behaviour which are used in a guidance service system deployed at a shopping mall, museum, or exhibition to serve multiple human users. Most such work consider the individual task’s cost change brought by environment changes, e.g., in [9] the optimal allocation is adjusted when associated costs change, while [7] deals with dynamic cost change using repeated auction, and [10] uses an incremental Hungarian method to repair original optimal assignment when costs change. H. Goldingay et al present a MRTA approach for a foraging robot swarm [6] which deals with tasks which are both spatially distributed and temporally sequential in a decentralised and self-organised manner, in which each agent uses local information to estimate both its own performance and its effect on the rest of the swarm. Another important but less mentioned aspect with dynamic MRTA is the runtime interaction between MRTA and each robot’s task scheduling or planning under different environment settings. B. Woosley et al [11] combines a greedy MRTA algorithm for task planning with a Field D*-based realtime path planning algorithm so that it is capable of handling dynamic changes in a robot’s path costs due to static/mobile obstacles. As far as we know, most current researches do not consider re-scheduling and re-allocating as well as their mutual interaction as a whole.

III. PROBLEM STATEMENT

For simplification, we assume: (1) Robots are homogeneous and have unlimited communication capabilities and physical resources. (2) Each robot has a schedule, i.e., a sequence of task points. (3) All tasks will be ultimately completed. (4) There are precedence constraints between task points. (5) Each robot can dynamically find the optimal path between locations.

The entire MRS receives tasks released at uncertain times. After release, tasks are distributed into the MRS and are then subject to rescheduling and reallocating. The set of all tasks is denoted as $T$, in which each $\tau \in T$ is a tuple $\langle \pi_0, \Pi, R \rangle$ where (1) $\pi_0 \in \Pi$ is a task point from which the execution of $\tau$ starts, (2) $\Pi$ is the set of task points which should be traversed during the execution of $\tau$, (3) $R \subseteq \Pi \times \Pi$ is the set of relations which determines the precedence constraints among task points in $\Pi$. Each task point $\pi$ is associated with a location $\lambda(\pi)$ which belongs to the dynamic entity related to $\pi$. Because of $R$, there are precedence constraints among task points, (even if these task points are allocated to different robots,) which result in cross-schedule dependencies and which must be respected during re-scheduling and re-allocating. In order to reflect such constraints, based on $R$, we define the precedence order $\sqsubset$ (and its inverse $\sqsubset$) on $\Pi$:

$$\forall \pi_1, \pi_2 \in \Pi[\pi_1 \neq \pi_2] :$$

$$\pi_1 \sqsubset \pi_2 \iff \exists \pi \in \Pi \setminus \{\pi_1, \pi_2\} : \pi_1 \sqsubset \pi \wedge \pi \sqsubset \pi_2$$

We use agents to model the robots in the MRS. The set of all agents is denoted as $A$, in which each $\alpha \in A$ is a tuple $\langle \sigma, \Omega, \pi_0 \rangle$, where (1) $\sigma$ is the current schedule of $\alpha$, which is a sequence of task points $\pi_1, ..., \pi_n$ to be performed by $\alpha$ in the future, and can change due to task release, task execution, re-scheduling and re-allocating of $\alpha$; (2) $\Omega$ is the set of auctions owned by $\alpha$ for re-allocating (see Section IV-B for detail); (3) $\sigma_0$ is the historical schedule (i.e., task points which have been completed by $\alpha$ in the order given by $\sigma_0$), which is preserved for analysing accumulated utility (see Section V-B for its use).

In order to evaluate the performance of a MRS, we establish a utility function based on the definition by Gerkey and Mataric [2], in which the utility of each robot $\alpha$ is defined as the sum of cost (i.e., negative utility) and quality (i.e., positive utility) of completing the schedule $\sigma$ of $\alpha$, i.e.,

$$||\alpha|| = ||\sigma|| = ||\alpha||_+ - ||\alpha||_-$$

where $||\alpha||_+$ (also $||\sigma||_+$) and $||\alpha||_-$ (also $||\sigma||_-$) are the quality and cost of completing the schedule $\sigma$, respectively. The single robot’s utility is the main target function of re-scheduling.

The cost (or negative utility) of each agent $\alpha$ is defined as the total cost of visiting all the task points in $\sigma$, i.e.,

$$||\sigma||_- = \sum_{i=1}^{\sigma} ||\sigma(i-1), \sigma(i)||_-$$

where $||\sigma(i-1), \sigma(i)||_- \text{ denotes the cost (usually energy)}$ for $\alpha$ to travel from task point $\sigma(i-1)$ to $\sigma(i)$, and $\sigma(0)$ is a special task point corresponding to the current status of $\alpha$. The computation of $||\sigma(i-1), \sigma(i)||_- \text{ depends on the concrete path from } \sigma(i-1) \text{ to } \sigma(i) \text{ which is planned by } \alpha$.

The quality (or positive utility) of each agent $\alpha$ is defined as the sum of qualities of all task points in $\sigma$, i.e.,

$$||\sigma||_+ = \sum_{i=1}^{\sigma} ||\sigma(i)||_+$$

where for each $i \in [1, |\sigma|]$, $||\sigma(i)||_+$ denotes the quality of executing task point $\sigma(i)$, which is defined by

$$||\sigma(i)||_+ = \frac{||\sigma(i-1), \sigma(i)||_-}{||\tau(\sigma(i))||_-} \cdot \frac{K_q}{||\tau(\sigma(i))||_{\text{time}}}$$
where (1) the left term represents the contribution (in terms of the cost proportion of $\sigma(i)$) to the completion of task $\tau(\sigma(i))$ (i.e., the task which owns task point $\sigma(i)$), where $\|\sigma(i-1),\sigma(i)\|$ is the travelling cost from $\sigma(i-1)$ to $\sigma(i)$, and $\|\tau(\sigma(i))\|$ is the total cost of all task points in $\tau$. In order to find a better robot to which it can transmit each task for all the unexecuted task points in its current schedule, in order to make best-effort adaptation of the schedule (as shown above, $\pi$ is the set of task points possessed by $\pi$, $\pi$ is the current location of $\pi$, $\sigma$ is the schedule to which $\pi$ belongs, which can be different from $\sigma$); (2) the right term represents the reward to the completion of $\tau(\sigma(i))$, which is to the inverse of $\|\tau(\sigma(i))\|_{\text{time}}$ (i.e., the time span from the release of $\tau(\sigma(i))$ to its completion); (3) $K_{\sigma}$ is a parameter determining the weight of the quality in utility computation, which is made constant in this work. Such quality definition is based on the consideration that users usually expect high responsiveness.

The total utility of the MRS $A$ is defined as the sum of utilities of all agents in $A$, i.e.,

$$\|A\| = \sum_{\alpha \in A} \|\alpha\|$$

In order to measure the balance of $A$, we define the total deviation of $A$ as the sum of differences between agents, i.e.,

$$\delta(A) = \sum_{\alpha, \alpha' \in A[\alpha \neq \alpha']} \|\alpha\| - \|\alpha'\|$$

Replacing all the appearances of $\sigma$ into $\sigma_0$, i.e., considering the historical accumulated effects rather than the instantaneous estimated value, we can get another group of system performance indicators (see Section V-B for their use), which are written as $\|\bullet\|_{0+}$, $\|\bullet\|_{0-}$, $\|\bullet\|_{0}$, $\delta_0(\bullet)$, etc. For example, the accumulated positive utility of an agent $\alpha$ is

$$\|\alpha\|_{0+} = \sum_{i=1}^{\|\sigma_0\|} \|\sigma_0(i)\|$$

The objective of our dynamic MRTA approach is to (1) maximise $\|A\|$, the total utility of the MRS, and (2) minimise $\delta(A)$, the deviation of individual utility among robots, i.e.,

$$\max \|A\|, \min \delta(A)$$

IV. APPROACH

In this section we present our approach to dynamic MRTA in following two sections. Section IV-A shows re-scheduling at intra-robot level, which applies a greedy algorithm that repeatedly makes best-effort adaptation of the current schedule of each robot, in order to achieve the higher estimated utility of that robot. Section IV-B shows re-allocating at inter-robot level, which allows each robot to repeatedly conduct auctions for all the unexecuted task points in its current schedule, in order to find a better robot to which it can transmit each task point so as to improve the estimated pairwise total utility (and ultimately improving the total utility of entire MRS).

A. Re-Scheduling at Intra-Robot Level

Fig. 2 shows the algorithm for adapting current schedule. For each robot, this algorithm will be periodically invoked in order to make best-effort improvement to $\sigma$. At each run, this algorithm will traverse entire $\sigma$; and for each task point $\sigma(j)$, it will try to shift $\sigma(j)$ to another index in $\sigma$ in order to achieve certain improvement in $\|\sigma\|$. This process will end until all the task points in $\sigma$ have been tried once, or when the algorithm can no longer improve $\|\sigma\|$.

Lines 1-4 initialise the PRE and POST functions, which are used to restrain the range within which a task point can be shifted. For each index $j = 1, \ldots, |\sigma|$, PRE($j$) and POST($j$) respectively defines the lower and upper limit for shifting $\sigma(j)$, i.e., $\sigma(j)$ can only move within (PRE($j$), POST($j$)) $\cap Z$, which ensures that the $\sigma$ always satisfies precedence constraints.

Line 5 sorts the indexes in $[1, |\sigma|] \cap Z$ (and then stores them into $Q$) according to a priority function $p: \Pi \mapsto \mathbb{R}$: for each $\pi \in \sigma$, $p(\pi)$ returns the priority of $\pi$ to be adjusted in the current run of re-scheduling, which is defined by

$$p(\pi) = \sum_{\mu_{mv} \in \pi} \exp(-d(\pi, \mu_{mv}))$$

where (1) $\Pi_{mv}$ is the set of task points possessed by $\alpha$ which, as shown in Fig. 3, during the time interval from last and this re-scheduling, have undergone location changes (due to the movement of the related dynamic entity) or have been inserted/removed to/from $\sigma$; (2) $d(\pi, \mu_{mv})$ is the composite distance between task points $\pi$ and $\mu_{mv}$, which is defined as

$$d(\pi, \mu_{mv}) = L \cdot |\lambda(\pi) - \lambda(\mu_{mv})| + |\text{IDX}(\pi, \sigma) - \text{IDX}(\mu_{mv}, \sigma)|$$

where (1) $L$ is an adjustable parameter, (2) for each task point $\pi$, $\lambda(\pi)$ is the current location of $\pi$, and (3) $\text{IDX}(\pi, \sigma)$ is the current index of $\pi$ within $\sigma$. As shown above, $p$ assigns higher values to the task points which are nearer (both in environment and in $\sigma$) to the task points that have been affected by the environment changes and/or re-allocating. It is expected that sorting by $p$ can accelerate the convergence of re-scheduling.

Lines 6-15 (the while loop) iteratively attempts to adapt $\sigma$ until there is little room for improvement. The loop picks one index $j$ from $Q$ and tries to improve $\|\sigma\|$ by shifting $\sigma(j)$ for a distance $d$ which satisfies $j + d \in \text{PRE}(j), \text{POST}(j) \cap Z$. The function $\nu_j$ represents the effect of shifting $\sigma(j)$ for a
Fig. 3. Reflecting re-allocating and environment changes on re-scheduling

distance $d$. Assume at $i^{th}$ iteration (where the schedule adapted by previous iterations is $\sigma_{i-1}$, the algorithm attempts to find the best adaptation $\pi_{j_{\text{max}}}^d$ of $\sigma_{i-1}$ so that $\sigma_i = \pi_{j_{\text{max}}}^d \sigma_{i-1}$ and the adaptation $\pi_{j_{\text{max}}}^d$ maximises the improvement indicator $||\pi_i^d \sigma|| - ||\sigma||$. If the algorithm succeeds in finding $\pi_{j_{\text{max}}}^d$, then it will apply this adaptation to $\sigma_{i-1}$ to generate $\sigma_i$, remove $j_{\text{max}}$ from $Q$ (Line 10), and update indexes (Lines 10-12) in $Q$, PRE, and POST, using the displacement function $\gamma^d_j$ (which satisfies $(\pi_i^d \sigma)(\gamma^d_j(i)) = \sigma(i)$ for $i \in [1, ||\sigma||] \cap \mathbb{Z}$; otherwise, it will quit the loop (Line 13).

B. Re-Allocating at Inter-Robot Level

Each agent $\alpha \in A$ possesses a set of auctions $\alpha \Omega$ which corresponds to the task points that can be reallocated by $\alpha$ to other robots. Each auction $\omega \in \alpha \Omega$ is a tuple $(\alpha, \pi, t_n, n)$, where (1) $\alpha$ is the robot which owns $\omega$, (2) $\pi$ is the task point encapsulated by $\omega$, (3) $t_n$ is the time at which $n^{th}$ reallocating of $\pi$ is done, (4) $n$ is the number of times that $\pi$ has been reallocated. After having been reallocated for $n$ times, $\pi$ is re-allocated only after time $t_{n+1}$, which is determined by

$$t_{n+1} = t_n + \lceil K \cdot n^M \rceil$$

where $K \in \mathbb{R}^+$ and $M \in \mathbb{R}^*$ are parameters (constant in this work) which together determine how the re-allocating frequency is updated. We can use this formula to restrain the amount of auctions performed by a MRS over time, thereby reducing the messages exchanged between robots and also the overhead for processing these messages.

Algorithm II Re-Allocating

\begin{algorithm}
input: Agent $\alpha$, $\omega \equiv \langle \alpha, \pi, t_n, n \rangle \in \alpha \Omega$
1. $\alpha \Omega \leftarrow \alpha \Omega \setminus \{\omega\}$
2. $\text{SEND}(\omega, A \setminus \{\alpha\})$
3. $c \leftarrow \text{new Clock}(\Delta t)$
4. $B_\omega \leftarrow \text{Receive}(c)$
5. $B_\omega \leftarrow \{\beta = \langle \alpha', \pi \rangle \in B_\omega : \alpha' \in A \setminus \{\alpha\} \land \phi(\beta) > 0\}$
6. $\beta_{\text{max}, \omega} \leftarrow \text{arg max } \phi(\beta)$
7. if $\beta_{\text{max}, \omega} = \langle \alpha, \alpha', \pi \rangle \neq \perp :$
8. $\text{SEND}(\perp, \alpha')$
9. else :
10. $\alpha \Omega \leftarrow \alpha \Omega \cup \{\omega\}$
11. end if
output: Best bid $\beta_{\text{max}, \omega}$
\end{algorithm}

Fig. 4. Algorithm used by each robot to conduct an auction for a task point

Fig. 4 shows the process of auctioning a task point $\pi$ which is currently owned by robot $\alpha$ to other robots. This algorithm will be triggered for each $\pi \in \alpha \Omega$ when $t = t_{n+1}$, where $t$ is the current system time, and $t_{n+1}$ is the time when $\pi$ can be re-allocated. After being triggered, the algorithm will try to reallocate $\pi$ to each robot $\alpha' \neq \alpha$ which is considered by $\alpha$ to be the fittest one (in terms of the expected improvement of such reallocation to the “pairwise” total utility of $\alpha$ and $\alpha'$) to execute $\pi$. The global effect of every robot executing such algorithm is that all task points which have not been executed will travel between different robots until it finds the fit one, and each step of such travelling will hopefully bring a small improvement to the total utility of the MRS.

At Line 2, robot $\alpha$ notifies every robot $\alpha' \neq \alpha$ about the beginning of auction $\omega$. After that, in Line 3-4, each robot $\alpha' \neq \alpha$ will respond to $\alpha$ with a bid, in the hope that it will be chosen by $\alpha$ to own $\pi$. The set of all bids returned by other robots is included in the set $B_\omega$, which is defined by

$$B_\omega = \{\beta = \langle \alpha, \alpha', \pi \rangle : \alpha' \in A \setminus \{\alpha\} \land \phi(\beta) > 0\}$$

where for each bid $\beta \in B_\omega$, $\phi(\beta)$ returns the fitness of $\beta$, which is defined as the increment in the pairwise total utility of $\alpha$ and $\alpha'$ after reallocating $\pi$ to $\alpha'$, i.e.,

$$\phi(\beta) = \Delta_\beta v(\alpha, \alpha') = v \circ \varsigma_\beta(\alpha, \alpha') - v(\alpha, \alpha')$$

where for each pair of agents $\alpha$ and $\alpha'$, their pairwise total utility $v(\alpha, \alpha')$ is defined as the total utility of $\alpha$ and $\alpha'$, i.e.,

$$v(\alpha, \alpha') = ||\alpha|| + ||\alpha'||$$

and $\varsigma_\beta : A \times A \rightarrow A \times A$ is an operator which represents the results of applying re-allocating (and re-scheduling after that, if re-scheduling is used) on both $\alpha$ and $\alpha'$. One important reason of considering pairwise rather than global total utility ($||A||$) is that: in the MRS, many auctions run concurrently with $\omega$, which may all affect the global task allocation; even if other robots are considered in the computation of the fitness in $\omega$, their respective utilities may soon undergo significant changes due to their own re-allocating, thereby making the fitness of $\omega$ (based on $||A||$) unable to accurately reflect the effects of $\omega$. Therefore we choose pairwise total utility for computing fitness of $\omega$, in the hope that it can give a limited, local, but more reliable estimation of the improvement by $\omega$.

In order to determine $\phi(\beta)$ for each bid $\beta$, both $\alpha$ and $\alpha'$ (which sends $\beta$ to $\alpha$) need to simulate the action of transmitting task point $\pi$ from $\alpha$ to $\alpha'$, due to the possible dependency between schedules of $\alpha$ and $\alpha'$. After all bids are checked by $\alpha$, the bid with the highest fitness value, $\beta_{\text{max}, \omega}$, will be chosen (Line 6), and the corresponding agent $\beta_{\text{max}, \omega}$ will be awarded with task point $\pi$, as shown in Line 8.

V. Experiments

We have conducted two groups of experiments (using a simple simulator developed by ourselves) in order to evaluate our MRTA approach. Group I (Fig. 5 and Table I) concerns how the approach behaves instantaneously and how such behaviour improves the performance indicators: total deviation and total utility of the MRS. Group II (Table II) concerns the effect of applying re-scheduling and re-allocating on the accumulated values of the performance indicators. In Group II we don’t take utility $||A||$ as a whole but separately examine cost $||A||_-$ (to be minimised) and quality $||A||_+$ (to
be maximised), in order to observe the accumulated effect of our approach in more detail. However in Group I we simply examine $\|A\|$ since it is the target function directly considered in both re-scheduling and re-allocating.

Assume (1) the MRS has 10 robots, (2) 100 tasks are periodically released with interval $= \Delta t_{rls}$ time units, (3) the maximum speed of dynamic entities $= v_{max}$. Both Group I and II concern the general effects of the environment parameters, $\Delta t_{rls}$ (directly related to the pressure of task execution on the MRS) and $v_{max}$ (reflecting how dynamic the environment is), to our MRTA approach. For the cases with re-allocating, we fix re-allocating-related parameters as $K = 1$ and $M = 1.2$.

A. Group I: Instantaneous Estimated Values

Fig. 5 shows the adjustment curves of instantaneous estimated value of $\|A\|$ and $\delta(A)$ under different settings ($\Delta t_{rls} \in \{100, 50\}$ and $v_{max} \in \{0, 0.5\}$) with re-scheduling (applied or not) and re-allocating (applied or not). Each adjustment curve can be divided into two parts by the point $t = [T] \cdot \Delta t_{rls}$, which is especially obvious for cases with higher $\Delta t_{rls}$ and/or $v_{max}$ values. The characteristics of an adjustment curve can be observed generally through: (1) the height (of the peak) of the curve, often reached at $t = [T] \cdot \Delta t_{rls}$, (2) overall slope of the curve, (3) oscillation of the curve, showing the intensity of instantaneous adjustment. Section $t < [T] \cdot \Delta t_{rls}$ corresponds to the gathering of tasks in the MRS (with task execution slowly in progress, but with slight effect in preventing the precipitous raise of the curves), while section $t > [T] \cdot \Delta t_{rls}$ shows the consumption of existing tasks (without accepting new ones).

Table I shows the adjustment effect on the average (over time) instantaneous estimated values of $\delta(A)$ and $\|A\|$, where $\hat{\delta}(\bullet) = \|\bullet\|_{\|\cdot\|_1} / \|\bullet\|_{\|\cdot\|_\infty}$, in which $\mu$ means the average (over time), $\perp$ (or $\|\cdot\|_\infty$) means re-scheduling and/or re-allocating is (or is not) applied, and the fraction is so arranged because we expect both $\delta(A)$ and $\|A\|$ (which is dominated by cost) to be small. Observing Table I and Fig. 5, we can see that: (1) both re-scheduling and re-allocating can flatten and lower the curves, thereby reducing average (over time) instantaneous estimated values; (2) re-allocating appears to have much higher such capability than re-scheduling, and bring more violent oscillation to the curves than re-scheduling, which implies its stronger global adjustment effect; (3) combinatorial case can produce curves with the characteristics (in height and slope) of both re-allocating only and re-scheduling only cases, and hence yield higher improvement rates than both cases; (4) the general influence of decreasing $\Delta t_{rls}$ and/or increasing $v_{max}$ on the curves is that the gradients of the curves become greater and the oscillation (due to adjustment) becomes more lenient; (5) the effect of increasing $v_{max}$ has more significant effect than decreasing $\Delta t_{rls}$, and thus more radically reduce the power of the re-scheduling and re-allocating in flattening and lowering the curves (in which re-allocating suffers most seriously, while the combinatorial case suffers less, and even can approximate the $v_{max} = 0$ case).

<table>
<thead>
<tr>
<th>Re-Scheduling</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-Allocating</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta t_{rls}$</th>
<th>$v_{max}$</th>
<th>$\delta(A)$</th>
<th>$|A|$</th>
<th>$\delta(A)$</th>
<th>$|A|$</th>
<th>$\delta(A)$</th>
<th>$|A|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>5.104</td>
<td>5.699</td>
<td>51.791</td>
<td>80.693</td>
<td>93.364</td>
<td>155.074</td>
</tr>
<tr>
<td>100</td>
<td>0.2</td>
<td>7.732</td>
<td>6.759</td>
<td>43.532</td>
<td>41.129</td>
<td>76.466</td>
<td>127.877</td>
</tr>
<tr>
<td>80</td>
<td>0.2</td>
<td>4.108</td>
<td>4.906</td>
<td>34.199</td>
<td>59.344</td>
<td>52.779</td>
<td>100.341</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>5.071</td>
<td>4.369</td>
<td>17.594</td>
<td>19.215</td>
<td>32.587</td>
<td>55.156</td>
</tr>
<tr>
<td>60</td>
<td>0.2</td>
<td>8.602</td>
<td>5.821</td>
<td>13.146</td>
<td>7.258</td>
<td>43.937</td>
<td>52.218</td>
</tr>
</tbody>
</table>

B. Group II: Accumulated Values

Table II shows the overall effect of adjustment on accumulated (over time) value of $\delta(A)$, $\|A\|_{\|\cdot\|_1}$, and $\|A\|_{\|\cdot\|_\infty}$, where $\hat{i}\delta(A) = \delta_0(A)_{\|\cdot\|_1} / \delta_0(A)_{\|\cdot\|_\infty}$, $\hat{i}\|A\|_{\|\cdot\|_1} = \|A\|_{\|\cdot\|_1} / \|A\|_{\|\cdot\|_\infty}$, and $\hat{i}\|A\|_{\|\cdot\|_\infty} = \|A\|_{\|\cdot\|_\infty} / \|A\|_{\|\cdot\|_1}$, in which $\perp$ (or $\|\cdot\|_\infty$) means re-scheduling and/or re-allocating is (or is not) applied. In the expression of $\hat{i}\|A\|_{\|\cdot\|_1}$, the $\perp$ term is taken as dividend because we expect larger $\|A\|_{\|\cdot\|_\infty}$. Compared to Group I, the improvement rates of Group II are obviously lower, and show a different way of changing with $\Delta t_{rls}$ and $v_{max}$. Observing Table II, we can see that: (1) re-allocating has significantly higher effect in improving $\delta_0(A)$ than re-scheduling; (2) re-allocating and re-scheduling have similar effects in improving $\|A\|_{\|\cdot\|_1}$, which show little change when $\Delta t_{rls}$ decreases, but show significant improvement when $v_{max}$ increases (due to the extremely poor behaviour of the control group, when both are not applied); (3) combinatorial case shows that: (i) its capability in improving $\delta_0(A)$ is close to the one in re-allocating only case, (ii) its capability in improving $\|A\|_{\|\cdot\|_1}$ is slightly higher than re-allocating only and re-scheduling only cases, (iii) its capability in improving $\|A\|_{\|\cdot\|_\infty}$ is significantly higher (especially when compared to re-scheduling only case), but is not boosted significantly when $v_{max}$ increases; (4) the capability of both re-allocating and re-scheduling (and thus their combination) in improving $\delta_0(A)$ is significantly improved when $v_{max}$ increases; (5) in contrast to Group I, decreasing $\Delta t_{rls}$ has no significant effect on the improvement rate, however, increasing $v_{max}$ significantly improves the improvement rate; (6) the capability in improving $\|A\|_{\|\cdot\|_\infty}$ of both re-scheduling and re-allocating shows no significant change when $\Delta t_{rls}$ decreases and/or $v_{max}$ increases.

VI. CONCLUSION

In this paper we have presented an approach to dynamic multi-robot task allocation which combines re-scheduling at intra-robot level with re-allocating at inter-robot level in order to continuously adjust the local and global utilities. We have conducted two groups of experiments in order to investigate the capability of the approach under different environment settings. The combination of re-scheduling and re-allocating generates significantly higher improvement to the total deviation and total utility of the MRS than re-scheduling only and re-allocating only cases.

We are developing an integrated dynamic MRTA approach which combines re-scheduling, re-allocating, perception, and
task execution, which we plan to deploy on a real world service robot system. In future works, we will introduce more realistic constraints and assumptions associated with specific application scenarios, and will use reinforcement learning to improve the performance of our approach under uncertain and partially known environments.

REFERENCES


FIG. 5. Adjustment curves of instantaneous estimated value of $\|A\|$ and $\delta(A)$ under different environment, re-scheduling and re-allocating settings

TABLE II

<table>
<thead>
<tr>
<th>Re-Scheduling</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t_{\text{cls}}$</td>
<td>$v_{\text{max}}$</td>
<td>$t \delta(A)$</td>
<td>$t|A|$</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>2.298</td>
<td>2.119</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>8.47</td>
<td>3.65</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>10.103</td>
<td>3.813</td>
</tr>
<tr>
<td>100</td>
<td>0.2</td>
<td>3.806</td>
<td>2.534</td>
</tr>
<tr>
<td>80</td>
<td>0.2</td>
<td>19.451</td>
<td>5.254</td>
</tr>
<tr>
<td>60</td>
<td>0.2</td>
<td>19.451</td>
<td>5.254</td>
</tr>
</tbody>
</table>