Improving Redundancy Availability: Dynamic Subtasks Modulation for Robots with Redundancy Insufficiency

Lu Chen*[1], Lipeng Chen*[2], Xiangchi Chen[1], Yi Ren[2], Longfei Zhao[2], Yue Wang[1], Rong Xiong[1]

Abstract— **This work presents an approach for robots to suitably carry out complex applications characterized by the presence of multiple additional constraints or subtasks (e.g. obstacle and self-collision avoidance) but subject to redundancy insufficiency. The proposed approach, based on a novel subtask merging strategy, enforces all subtasks in due course by dynamically modulating a virtual secondary task, where the task status and soft priority are incorporated to improve the overall efficiency of redundancy resolution. The proposed approach greatly improves the redundancy availability by unitizing and deploying subtasks in a fine-grained and compact manner. We build up our control framework on the null space projection, which guarantees the execution of subtasks does not interfere with the primary task. Experimental results on two case studies are presented to show the performance of our approach.**

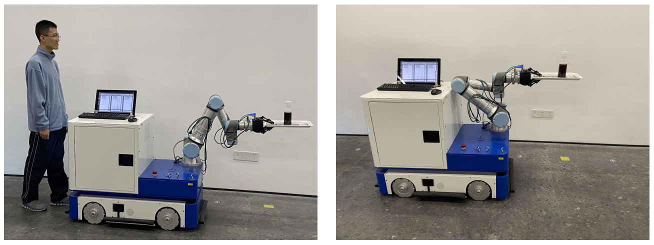

(a) Human avoidance subtask (b) Self-collision avoidance subtask

Fig. 1. A drink-serving robot dynamically allocates relatively insufficient redundancies to accomplish multiple subtasks while serving drinks. (a) The human avoidance subtask takes redundancies as the human walks close. (b) The self-collision subtask takes redundancies once the human is far away.

I. Introduction

Redundant robots have been dominating with growing popularity the robotics community in virtue of their increased dexterity, versatility and adaptability [1], [2], [3]. However, except for few highly specialized systems, most redundant robots still underperform due to lack of relatively sufficient redundancies, especially when operating in unstructured or dynamic environments like households or warehouses characterized by the occurrence of multiple additional subtasks. Take a drink-serving task as illustrated in Fig. 1 for example. Even though the mobile robot is already equipped with nine degrees of freedom (DOF), as the robot carries a tray upright to serve drinks, only three DOFs will be left as redundancies. However, besides the primary serving task, the robot is frequently confronted with a large number of additional constraints or subtasks, e.g. obstacles, walking humans and singularity avoidance, which may actually require far more redundancies than the remaining ones. That is, the robot may not be able to deal with all subtasks simultaneously due to the lack of redundancies for subtasking.

We focus on the constrained scenario of *redundancy resolution* problems [4], [5], [6] like this, where a redundant robot is supposed to carry out a primary task accompanied by multiple additional subtasks **but** subject to redundancy insufficiency.

A straightforward engineering way out of the above redundancy dilemma is to introduce more kinematic redundancies into the robot mechanical structure, which apparently is way too expensive to be repeatable. The majority of prior works on redundancy resolution, either via optimization [7], [8], [9] or task augmentation [10], [11], [12], however, are fundamentally under the premise the robot can provide sufficient redundancies i.e. all subtasks can be performed simultaneously with required redundancies.

Rather, we noticed that in fact not all aforementioned subtasks have to be performed simultaneously or synchronously thanks to task feature and environment characteristics[1]. For example, a whole-course obstacle avoidance subtask can actually be idle during most of the runtime until some obstacle appears within a certain threshold region, and therefore can be deferred from taking redundancy. Such characteristics give rise to the potential of asynchronicity among subtasks, which essentially accommodates most practical robot applications characterized by dynamic and unstructured environments.

It leads to a lightweight but effective solution that the robot can dynamically allocate redundancies to subtasks according to some common rules like task urgency, activeness and importance. For example in Fig. 1, as the robot carries out the primary drinking-serving task, if a human moves closer to the robot (Fig. 1(a)), the subtask of human avoidance is of an increasing and ultimately dominating priority of taking all redundancies, while all other substasks will be temporarily frozen since no more redundancy is available. As the human walks away, the robot will eventually release the (part of) redundancies, until some other subtask takes them, e.g. the self-collision avoidance subtask (Fig. 1(b)).

In this work, we borrow ideas from asynchronous time-division multiplexing (ATDM), propose an approach to subtask management for redundant robots subject to redundancy insufficiency. Our approach unfolds as follows: we first unitize all multi-dimensional subtasks to be executed along

*First two authors contributed equally to this work.

[1]Lu Chen, Xiangchi Chen, Yue Wang and Rong Xiong are with Zhejiang University, Zhejiang, China. {lu-chen, chenxiangchi, ywang24, rxiong}@zju.edu.cn

[2]Lipeng Chen, Yi Ren and Longfei Zhao are with Tencent, China. {lipengchen,evanyren,longfeizhao}@tencent.com

[1]It is acknowledged that there exist cases where subtasks need to be performed strictly simultaneously. For such cases, the engineering augmentation as aforementioned is the only solution.

with the primary task into a set of one-dimensional *elementary subtasks*. This step allows us to greatly improve the redundancy availability by deploying subtasks in a more fine-grained and compact manner. We then manage elementary subtasks by fusing them into a virtual multi-dimensional *secondary task* w.r.t. the primary task. We propose a novel subtask merging operator and an efficient updating strategy to dynamically modulate the secondary task in compliance with the *task status* and *soft priority* derived heuristically. Based on the approach, all subtasks can be suitably performed in due course.

Our control framework is built upon previous work of task priority based redundancy resolution [1], [4], [5], which guarantees the low-level tasks executed in the null space do not interfere with the high-level tasks. We integrate our subtask merging strategy into the null space projection technique to derive a general control framework of subtask management for redundant robots subject to redundancy insufficiency. In this framework, the primary task is perfectly performed using a number of required DOFs, while all other subtasks are suitably carried out as a virtual dynamic secondary task using the remaining insufficient redundancy, but without affecting the primary task.

The paper is organized as follows. Sec. II and III reviews and recapitulates prior related works. Sec. IV presents details of our approach to manage multiple subtasks subject to redundancy insufficiency. Sec. V introduces two case studies with experimental results to verify the performance of our approach. Sec. VI concludes this paper and our future work.

II. Related Work

Our work is in the intersection of *inverse kinematic control*, *redundancy resolution* and *prioritized multitasking*.

The very early works of redundant robots have derived the fundamental solution to redundancy resolution by using Jacobian pseudoinverse to find the instantaneous relationship between the joint and task velocities. The later extensive investigations, essentially, have been developed explicitly or implicitly from the Jacobian pseudoinverse via either optimization or task augmentation. Typically, redundancy resolution via optimization incorporates additional additional subtasks or objectives by minimizing certain task-oriented criteria [6], [7]. For example, obstacle avoidance is enforced by minimizing a function of artificial potential defined over the obstacle region in configuration space [8]. The task augmentation approaches address additional subtasks by augmenting an integrated task vector containing all subtasks, where the extended or augmented Jacobians are formulated to enforce additional tasks [10], [11], [12].

The majority of frequently applied approaches to redundancy resolution are fundamentally based on the null space projection strategy [13], [14], [15]. In compliance with a dynamically consistent task hierarchy of this line of work, additional subtasks are preformed only in the null space of a certain higher-priority task, typically by successive null space projections [14], [16] or augmented null space projections [17], [18]. We also build our control law upon this technique by performing all subtasks in the null space of the primary task. The aforementioned Jacobian pseudoinverse centered approaches, however, work mostly under the premise of sufficient redundancies for multitasking, which instead is the major challenge motivating and addressed by our work.

Our work is also related to prioritized multitask control, which is mainly focused on addressing task incompatibility by defining suitable evolution of task priorities [19], [20], [21]. Typically, priorities are given to safety-critical tasks such as balancing if conflict or incompatibility occurs [22], [23]. Different from this line of studies, our work mainly focus on the issue of insufficient robot redundancy, and therefore all substasks have to compete for redundancy even in the absence of task incompatibility.

III. Background

Our work is built upon prior literature in inverse differential kinematics and null space projection based redundancy resolution.

A. Inverse Differential Kinematics

Let $\boldsymbol{q} \in \mathbb{R}^n$ denote the joint configuration of a robot with n degrees of freedom. Let $\boldsymbol{x} \in \mathbb{R}^m$ denote the vector of task variables in a suitably defined m-dimensional task space. The first-order *differential kinematics* is usually expressed as

$$\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{1}$$

where $\dot{\boldsymbol{x}}$, $\dot{\boldsymbol{q}}$ are vectors of task and joint velocities respectively. $\mathbf{J}(\boldsymbol{q})$ is the $m \times n$ Jacobian matrix. The dependence on $\boldsymbol{q}$ is omitted hereafter for notation compactness.

Typically, one has $n \geq m$ for a redundant robot, i.e. the robot has a $(n-m)$-dimensional *redundancy space* for subtasking. Then the general *inverse differential kinematics* solution of Eq. 1 is usually expressed as

$$\dot{\boldsymbol{q}} = \mathbf{J}^{+}\dot{\boldsymbol{x}} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\dot{\boldsymbol{q}}_0 \tag{2}$$

where $\mathbf{J}^{+} \in \mathbb{R}^{n \times m}$ is the pseudoinverse matrix of $\mathbf{J}$. $\mathbf{N}(\mathbf{J}) = \mathbf{I} - \mathbf{J}^{+}\mathbf{J} \in \mathbb{R}^{n \times n}$ is an operator projecting any arbitrary joint velocity $\dot{\boldsymbol{q}}_0 \in \mathbb{R}^n$ into the null space of $\mathbf{J}$, i.e. the robot redundancy space.

B. Null Space Projection based Redundancy Resolution

The projection of $\dot{\boldsymbol{q}}_0$ onto the null space ensures no effect on the primary task. Under this premise, the early works [1], [4], [5] have proposed the control framework of redundancy resolution with task priority, which essentially consists of computing a $\dot{\boldsymbol{q}}_0$ to suitably enforce a secondary task in the null space of the primary task.

With reference to Eq. 2, the inverse kinematics solution considering a two-order of task priorities (indexed by 1, 2 for the primary and secondary task respectively) can then be expressed as

$$\dot{\boldsymbol{q}} = \mathbf{J}_1^{+}\dot{\boldsymbol{x}}_1 + (\mathbf{I} - \mathbf{J}_1^{+}\mathbf{J}_1)[\mathbf{J}_2(\mathbf{I} - \mathbf{J}_1^{+}\mathbf{J}_1)]^{+}(\dot{\boldsymbol{x}}_2 - \mathbf{J}_2\mathbf{J}_1^{+}\dot{\boldsymbol{x}}_1) \tag{3}$$

where $\dot{\boldsymbol{x}}_1$, $\dot{\boldsymbol{x}}_2$ and $\mathbf{J}_1$, $\mathbf{J}_2$ are the task velocities and Jacobian matrices of the primary and secondary task respectively.

As illustrated in Fig. 2, we build our control framework upon Eq. 3, where we model a virtual dynamic secondary

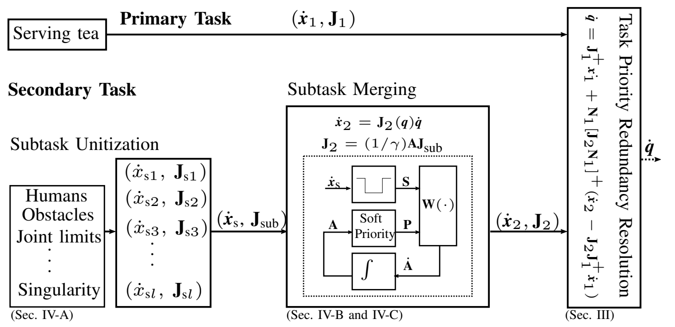

Serving tea
Primary Task
$(\dot{\boldsymbol{x}}_1, \mathbf{J}_1)$
Task Priority Redundancy Resolution
$\dot{\boldsymbol{q}} = \mathbf{J}_1^+\dot{\boldsymbol{x}}_1 + \mathbf{N}_1[\mathbf{J}_2\mathbf{N}_1]^+(\dot{\boldsymbol{x}}_2 - \mathbf{J}_2\mathbf{J}_1^+\dot{\boldsymbol{x}}_1)$
$\dot{\boldsymbol{q}}$
Secondary Task
Subtask Merging
$\dot{x}_2 = \mathbf{J}_2(\boldsymbol{q})\dot{\boldsymbol{q}}$
$\mathbf{J}_2 = (1/\gamma)\mathbf{A}\mathbf{J}_{\text{sub}}$
Subtask Unitization
Humans
Obstacles
Joint limits
Singularity
$(\dot{x}_{s1}, \mathbf{J}_{s1})$
$(\dot{x}_{s2}, \mathbf{J}_{s2})$
$(\dot{x}_{s3}, \mathbf{J}_{s3})$
$(\dot{x}_{sl}, \mathbf{J}_{sl})$
$(\dot{\boldsymbol{x}}_s, \mathbf{J}_{\text{sub}})$
$\dot{\boldsymbol{x}}_s$
S
A
Soft Priority
P
$\mathbf{W}(\cdot)$
$\int$
$\dot{\mathbf{A}}$
$(\dot{\boldsymbol{x}}_2, \mathbf{J}_2)$
(Sec. IV-A)
(Sec. IV-B and IV-C)
(Sec. III)

Fig. 2. Overview of the Approach.

task for subtasks, and then deploy it in the null space of the primary task, such that all subtasks can be suitably executed as good as possible without disturbing the primary task.

IV. Method

This section presents our approach to manage multiple subtasks subject to redundancy insufficiency (Fig. 2).

A. Subtask Unitization

We first split and unitize all multi-dimensional subtasks to be executed along with the primary task into a set of one-dimensional *elementary subtasks*. For example, the obstacle avoidance of a mobile robot can be unitized into three elementary subtasks of *x*-direction, *y*-direction and *yaw*-rotation obstacle avoidance. In this manner, the subtasks can be unitized into a set of elementary subtasks expressed as

$$\dot{x}_{si} = f_i(\boldsymbol{\xi}_i) \in \mathbb{R} \quad i = 1, 2, ..., l \tag{4}$$

where l is the number of total elementary subtasks. $\boldsymbol{\xi}_i$ is a vector of all related parameters (i.e. the real robot state), and $\dot{x}_{si}$ is the desired velocity of the *i*-th elementary subtask. Each elementary subtask expressed in the form of Eq. 4 need to ensure global stability during construction. Note that the number of elementary subtasks can be less than or equal to the number of redundancies (i.e. $n - m \geq l$), which implies the robot can provide sufficient redundancies for subtasking. We focus on the opposite case ($n-m<l$), where the subtasks have to compete for redundancy due to insufficiency.

The subtask unitization allows our approach to deploy elementary subtasks in a more fine-grained and compact manner, and therefore improve the overall redundancy utilization and availability. Stacking all elementary subtasks together yields a *subtask vector*

$$\dot{\boldsymbol{x}}_s = [\dot{x}_{s1}\ \dot{x}_{s2}\ ...\ \dot{x}_{sl}]^{\mathrm{T}} = [f_1\ f_2\ ...\ f_l]^{\mathrm{T}} \tag{5}$$

Note that we associate an implicit order of elementary subtask priority by index in $\dot{\boldsymbol{x}}_s$, i.e. the smaller the index i, the higher the priority of its corresponding elementary subtask.

Suppose the first-order differential kinematics for the i-th elementary subtask is expressed as

$$\dot{x}_{si} = \mathbf{J}_{si}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{6}$$

where $\mathbf{J}_{si} \in \mathbb{R}^{1\times n}$ is its Jacobian matrix. Substituting Eq. 6 into Eq. 5 yields

$$\dot{\boldsymbol{x}}_s = \mathbf{J}_{\text{sub}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{7}$$

where $\mathbf{J}_{\text{sub}} = [\mathbf{J}_{s1}^{\mathrm{T}}\ \mathbf{J}_{s2}^{\mathrm{T}}\ ...\ \mathbf{J}_{sl}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{l\times n}$ is the merged Jacobian matrix for the elementary subtask set.

B. Merging Subtasks into A Dynamic Secondary Task

We then build a *virtual secondary task* $\dot{\boldsymbol{x}}_2$ from the set of elementary subtasks $\dot{\boldsymbol{x}}_s$ in line with Eq. 3

$$\dot{\boldsymbol{x}}_2 = \mathbf{H}(\dot{\boldsymbol{x}}_s) \tag{8}$$

where $\mathbf{H}(\cdot)$ is an operator dynamically allocating $n-m$ robot redundancies to l elementary subtasks $\dot{\boldsymbol{x}}_s$ during runtime.

Multi-Subtask Merging Matrix: In order to construct the operator $\mathbf{H}(\cdot)$, we first define a multi-subtask merging matrix

$$\mathbf{A}(t) := \begin{bmatrix} \alpha_{11}(t) & \alpha_{12}(t) & \cdots & \alpha_{1l}(t) \\ \alpha_{21}(t) & \alpha_{22}(t) & \cdots & \alpha_{2l}(t) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{(n-m)1}(t) & \alpha_{(n-m)2}(t) & \cdots & \alpha_{(n-m)l}(t) \end{bmatrix} \tag{9}$$

where each entry α_{ij} denotes the weight of the i-th redundancy to be allocated to the j-th elementary subtask varying w.r.t. time. It satisfies $\sum_{j=1}^{l} \alpha_{ij} = \gamma$, where $\gamma \in [0.5, 1]$ is the upper bound for entries in $\mathbf{A}$. The dependence on t is omitted hereafter for notation compactness. The matrix is initialized with

$$\mathbf{A}_0 := \begin{bmatrix} \gamma \cdot \mathbf{I}_{(n-m)\times(n-m)} & \mathbf{0}_{(n-m)\times(l-n+m)} \end{bmatrix}$$

which implies the $n-m$ robot redundancies will be initially allocated to the first $n-m$ elementary tasks in $\dot{\boldsymbol{x}}_s$, in keeping with the aforementioned implicit indexing task priority.

Virtual Secondary Task: Then the virtual secondary task $\boldsymbol{x}_2$ is defined as a weighted contributions of l subtasks as

$$\begin{aligned} \dot{\boldsymbol{x}}_2 &= \mathbf{H}(\dot{\boldsymbol{x}}_s) = (1/\gamma)\cdot \mathbf{A}_{(n-m)\times l}\dot{\boldsymbol{x}}_{s(l\times 1)} = (1/\gamma)\sum_{j=1}^{l} \boldsymbol{\alpha}_j \dot{x}_{sj} \\ &= (1/\gamma)\begin{bmatrix} \sum_{j=1}^{l} \alpha_{1j}\dot{x}_{sj} & \sum_{j=1}^{l} \alpha_{2j}\dot{x}_{sj} & ... & \sum_{j=1}^{l} \alpha_{(n-m)j}\dot{x}_{sj} \end{bmatrix}^{\mathrm{T}} \\ &= [\dot{x}_{21}\ \ \dot{x}_{21}\ \ \ldots\ \ \dot{x}_{2(n-m)}]^{\mathrm{T}} \end{aligned} \tag{10}$$

where γ acts as a normalizing factor. Eq. 10 also implies at the i-th redundancy, the merging matrix $\mathbf{A}$ dynamically allocates a virtual task $\dot{x}_{2i}$ characterized by a weighted sum of l elementary subtasks.

Null Space Control: Substituting Eq. 7 and 10 into Eq. 1 yields

$$\dot{\boldsymbol{x}}_2 = \mathbf{J}_2(\boldsymbol{q})\dot{\boldsymbol{q}} = (1/\gamma)\mathbf{A}\mathbf{J}_{\text{sub}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{11}$$

where $\mathbf{J}_2 = (1/\gamma)\mathbf{A}\mathbf{J}_{\text{sub}}$ is the (merged) Jacobian matrix of the virtual secondary task. Then substituting Eq. 10 and 11 into Eq. 3 yields our law of redundancy resolution subject to insufficiency

$$\begin{aligned} \dot{\boldsymbol{q}} &= \mathbf{J}_1^+\dot{\boldsymbol{x}}_1 + \mathbf{N}_1\mathbf{J}_{\text{sub}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{J}_{\text{sub}}\mathbf{N}_1\mathbf{J}_{\text{sub}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}})^{-1}(\mathbf{A}\dot{\boldsymbol{x}}_s - \mathbf{A}\mathbf{J}_{\text{sub}}\mathbf{J}_1^+\dot{\boldsymbol{x}}_1) \\ \mathbf{N}_1 &= \mathbf{I} - \mathbf{J}_1^+\mathbf{J}_1 \in \mathbb{R}^{n\times n} \end{aligned} \tag{12}$$

which plays a fundamental role in our control framework. The next section explains how our algorithm dynamically modulates $\dot{\boldsymbol{x}}_2$ to manage subtasks under this framework.

C. Update of the Merging Matrix

With reference to Eq. 10−12, the dynamic control of multiple subtasks relies essentially on the update of $\mathbf{A}$. We formulate an updating strategy to proactively modulate the updating rate of $\mathbf{A}$ by incorporating *task status* and *soft priority* derived heuristically.

Task Status Matrix: We define a task status matrix $\mathbf{S}$ to modulate the updating rate in compliance with task status

$$\mathbf{S} = \mathrm{diag}(\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_l) \tag{13}$$

where $\bar{f}_i \in [0,1]$ quantifies the activation status of the i-th elementary subtask $\dot{x}_{\mathrm{s}i}$ with a normalized scalar. Specifically, if $\dot{x}_{\mathrm{s}i}$ arrives at a stable state, then there is $\dot{x}_{\mathrm{s}i} = 0, \bar{f}_i = 0$. That is, the *i*-th elementary subtask has been completed and there is no need to assign redundancy to it. On the contrary, if $\bar{f}_i \to 1$, it indicates the *i*-th elementary subtask is still active and therefore waiting be allocated with a redundancy.

Here we specify $\bar{f}_i$ with the normalizing function

$$\bar{f}_i = 1/(1 + e^{k_i(d_i + \dot{x}_{\mathrm{s}i})}) + 1/(1 + e^{k_i(d_i - \dot{x}_{\mathrm{s}i})}) \tag{14}$$

where k_i and d_i are the response slope and sensitivity range of the normalizing function. Note one can come up with some other definitions of task status, e.g. one considering the task amplitude. Here we treat all subtasks equally and focus on if an elementary subtask is completed or not.

Soft Priority Matrix: We derive a *soft priority matrix* $\mathbf{P}$ to proactively modulate the updating rate

$$\mathbf{P}(t) := \begin{bmatrix} p_{11}(t) & p_{12}(t) & \cdots & p_{1l}(t) \\ p_{21}(t) & p_{22}(t) & \cdots & p_{2l}(t) \\ \vdots & \vdots & \ddots & \vdots \\ p_{(n-m)1}(t) & p_{(n-m)2}(t) & \cdots & p_{(n-m)l}(t) \end{bmatrix} \tag{15}$$

where each entry $p_{ij} \in (0,1)$ implies a certain value of *soft priority* proactively modulating the updating rate of the weight α_{ij}.The soft priority is derived by the following rules[2]

$$p_{ij} = \prod_{u=0}^{i-1}(1-\alpha_{uj}) \prod_{v=0}^{j-1}(1-\alpha_{iv}) \prod_{u \neq i}(\gamma - \alpha_{uj}) \tag{16}$$

for $i = 1, 2, \ldots, (n-m)$ and $j = 1, 2, \ldots, l$. Each entry p_{ij} extracts implicit soft priority information from $\mathbf{A}$ by explicitly considering the weight distribution over its corresponding redundancy (i-th row) and elementary subtask (j-th column):

- The term $\prod_{u=0}^{i-1}(1-\alpha_{uj})$ indicates the updating rate of α_{ij} is affected by the weight distribution (for the *j*-th elementary subtask) over the $(i-1)$ redundancies previous to the current *i*-th one. Specifically, given a *j*-th elementary subtask, if its weight at any other redundancy (denoted as *u*-th) previous to the current *i*-th one is close to γ (i.e. $\alpha_{uj} \to \gamma$), it is more likely to be assigned to the *u*-th redundancy. Therefore, the weight at the current *i*-th redundancy will be relatively reduced to proactively quit the competition for the j-th elementary subtask. On the contrary, if its weight at any previous redundancy is close to zero, the weight at the current redundancy will be relatively raised proactively to improve the chance of winning.
- The term $\prod_{v=0}^{j-1}(1-\alpha_{iv})$ indicates, symmetrically, the updating rate of α_{ij} is affected by the weight distribution (at the i-th redundancy) over the $j-1$ elementary subtasks previous to the current *j*-th one. This term decides if the *j*-th elementary subtask should proactively quit or stay in the competition for the *i*-th redundancy.
- The term $\prod_{u\neq i}(\gamma-\alpha_{uj})$ acts as a redundancy keeper by rejecting or zeroing out the weight update at α_{ij} if the j-th elementary subtask has been allocated to any other redundancy (denoted as u-th and therefore $\alpha_{uj} = \gamma$) rather than the current i-th one. This guarantees the *j*-th elementary subtask will be kept in a redundancy once being allocated to and therefore would not jump back and forth among different redundancies.

The soft priority derived above is consistent with the aforementioned indexing priority by explicitly considering the weight distribution over previous redundancies and subtasks. It proactively tuning the updating rate and therefore leads to a faster convergence speed of the merging matrix $\mathbf{A}$. Such a prioritizing strategy is aimed at improving the efficiency of redundancy resolution, such that all elementary subtasks can be suitably performed in due course. Note one can come up with some other prioritizing strategies in accordance with context [21], [22], [23].

Updating the Merging Matrix: We define the updating rate $\dot{\mathbf{A}}$ as a combined effect of the task status $\mathbf{S}$ and the soft priority $\mathbf{P}$, and formulate it based on the winner-take-all strategy[3] $\mathbf{W}(\cdot)$

$$\dot{\mathbf{A}} = \mathbf{W}(\mathbf{P}, \mathbf{S}, \mathbf{A}) \tag{17}$$

Then the subtask merging matrix $\mathbf{A}$ is updated as follows

$$\mathbf{A}_{t+1} = \max(\mathbf{0}, \min(\gamma\mathbf{E}, \mathbf{A}_t + \dot{\mathbf{A}}_t \Delta t)) \tag{18}$$

where $\mathbf{E}$ is an all-ones matrix, and Δt is the update interval.

V. Experiment Results

This section presents two test cases followed by experimental results to show the performance of our approach.

A. Experimental Cases

I. Drink-Serving: As introduced previously in Fig. 1, the first test case is about a mobile robot serving drinks along a desired path. We implement this test case on a real six-DOF UR16e robot manipulator mounted on an omnidirectional mobile platform. Therefore, the robot has in total nine DOFs. The primary task of serving drinks requires six DOFs and therefore leaves three DOFs as redundancies for subtasking. The subtasks in this case involve:

[2]Note that a dummy row $\alpha_{0j} = 0,\ j = 1, 2, \ldots, l$ and a dummy column $\alpha_{i0} = 0,\ i = 1, 2, \ldots, (n-m)$ are added into $\mathbf{A}$ for the sake of expression simplicity. $\alpha_{0j} = 0$ implies the dummy 0-th redundancy will not be assigned to any subtask. $\alpha_{i0} = 0$ implies the dummy 0-th task will not occupy any redundancy.

[3]A detailed explanation of the algorithm and a proof of weight convergence are provided here: `https://github.com/AccotoirDeCola/WinnerTakeAll`.

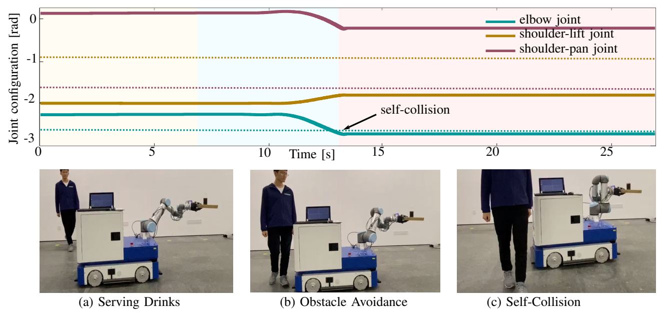


Fig. 3. The traditional approach: The robot collides with itself at the elbow joint (the blue line) at around 13s, as the self-collision subtask is not treated during the whole process due to redundancy insufficiency. Each solid line represents a relevant joint for self-collision avoidance, while the dotted line in the same color represents its joint-collision limit.

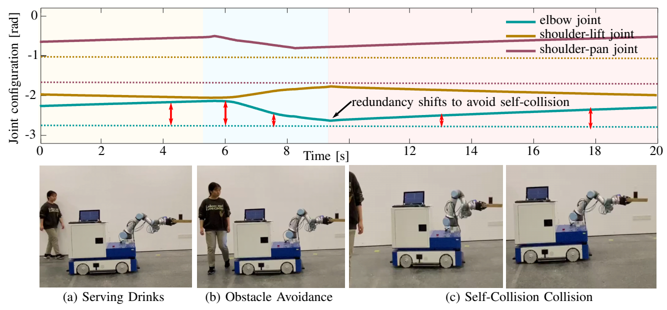


Fig. 4. Our approach: All subtasks are suitably performed. Three redundancies are first shifted to the obstacle-avoidance subtask to avoid the walking human at round 5s, and then given back to three elementary self-collision subtasks to avoid potential self-collision at around 9s.

- A three-dimensional *obstacle-avoidance* subtask, e.g. avoiding the walking human, which can be split into three elementary obstacle-avoidance subtasks.
- A three-dimensional *self-collision* avoidance subtask, e.g. avoiding the collision between the manipulator and the platform, which can be split into three elementary self-collision avoidance subtasks.

Ideally, both subtasks should be performed simultaneously along with the primary task. However, due to the lack of sufficient redundancies, the six elementary subtasks have to compete for three redundancies during runtime.

II. Circle-Drawing: As illustrated in Fig. 8, the second case is about a manipulator drawing a circle along a desired end-effector path. We implement this test case using the same robot as the first case, but the mobile platform is fixed at a certain location. Therefore, the robot has in total six DOFs. The primary task of circle drawing requires three DOFs and therefore leaves three DOFs as redundancies for subtasking. The subtasks in the case involve:

- A three-dimensional *singularity-avoidance* subtask, which can be split into three elementary singularity-avoidance subtasks.
- A one-dimensional *wrist-limit* subtask, which simply constraints the wrist joint to a desired angle.

Therefore, there are four elementary subtasks competing for three redundancies in this case.

B. Experimental Results

We test our approach (Eq. 12) on both cases and compare it with the traditional approach (Eq. 3). Briefly, given a case,

- The traditional approach first assigns a number of required DOFs to primary task. Then it allocates the remaining redundancies to as many subtasks as it can *and* then keep the redundancy allocation.
- The subtask-merging based approach (our approach), as explained in Sec. IV, first assigns the required DOFs to primary task. Then it dynamically allocates the remaining redundancies to all elementary subtasks generated from subtask unitization in due time.

I. Experimental Results of Drink-Serving: Fig. 3 and 4 show the results generated respectively by the traditional and our approach during the whole process of the self-collision avoidance subtask. Fig. 5 shows the redundancy shift among six elementary subtasks (i.e. the evolution of weights in **A**) generated by our approach.

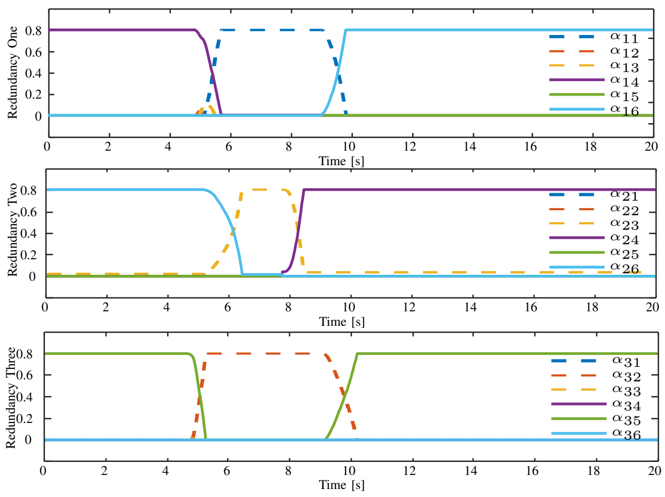
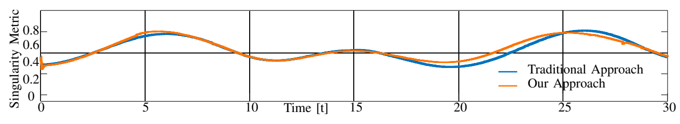


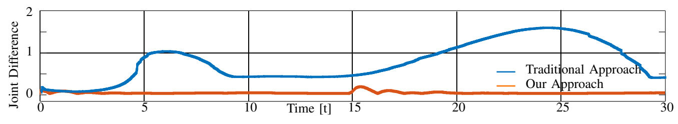
Fig. 5. Redundancies shift dynamically among elementary subtasks: Our approach dynamically allocates three redundancies to six elementary subtasks in due course. Each subfigure corresponds to a redundancy, where three dotted coloured lines correspond to the weights of three elementary obstacle-avoidance subtasks, and three solid coloured lines correspond to the weights of three elementary self-collision subtasks, i.e. the update of **A**.

In this case, as shown in Fig. 3, the traditional approach allocates three redundancies to the obstacle-avoidance subtask, and then leaves the self-collision subtask untreated since there is no more redundancy available. As a result, even though the moving human is successfully avoided during the whole course (as the obstacle-avoidance subtask is taking all redundancies), the robot collides with itself at the elbow joint at around 13 s and locks its manipulator henceforth for mechanical safety, i.e. the robot fails in executing the case.

Instead, as shown in Fig. 4 and 5, our approach dynamically allocates three redundancies to six elementary subtasks, and therefore all subtasks are suitably performed in due course. Specifically, the three redundancies are initially taken by the self-collision subtask, therefore the relative difference between each joint to its corresponding joint-collision limit (illustrated by the red double-arrowed line



(a) Performance on the Singularity-avoidance Subtask



(b) Performance on the Wrist-Limit Subtask

Fig. 6. Both approaches perform well in the singularity-avoidance subtask, while our approach outperforms in properly addressing the wrist-limit subtask.

segments in Fig. 4) increases in this phase (0s-5s). As the human enters the robot's sensing range of obstacle avoidance from around 5s, the redundancies are shifted to three self-collision elementary subtasks to keep the robot away from the walking human. Meanwhile, as a result, the joint differences for self-collision decrease (but not to zero) till around 9s, when the redundancies are shifted back to three elementary self-collision subtasks first and last. Accordingly, the joint differences increase again to avoid potential self collisions. All above redundancy shifts can be directly observed in Fig. 5.

Remarkably, Fig. 4 and 5 also show redundancy shifts do not need to happen simultaneously, even for the same subtask. That is, our approach allocates redundancies directly to one-dimensional elementary subtasks rather than their corresponding high-level multi-dimensional subtasks. This is thanks to the subtask unitization as introduced in Sec. IV-A, which greatly improves the redundancy availability and utilization. For example, from around 8s to 10s in Fig. 5, the second redundancy is shifted to an elementary self-collision subtask, while the other two redundancies are still occupied by two elementary obstacle-collision subtasks. It is also suggested from both figures that the redundancy shift can be performed swiftly (mostly within 1s) and smoothly by our approach.

II. Experimental Results of Circle-Drawing:

Fig. 6 shows results for the second case on the singularity-avoidance and wrist-limit subtasks generated by the traditional and our approach respectively. Both approaches perform well in the singularity-avoidance subtask (Fig. 6(a)) while the traditional approach underperforms in the wrist-limit subtask due to redundancy insufficiency (Fig. 6(b)).

Fig. 7 shows the redundancy shifts among four elementary subtasks (i.e. the evolution of **A**) generated by our approach. Specifically, from 0s to around 9s, two elementary singularity-avoidance subtasks and the wrist-limit subtask are performed. Then at around 9s, the second redundancy is shifted from one elementary singularity-avoidance subtask to the other, i.e. a redundancy shift happens between two elementary subtasks unitized from the same high-level subtask. This further proves that our approach allocates redundancies in the elementary subtask level. Such a redundancy shift is in fact due to the change of task status, i.e. a (nearly) completed subtask gives its redundancy to an alive subtask.

Remarkably, Fig. 7 shows the primary task can be performed well by both approaches, i.e. the primary task is not affected by the execution of subtasks. This is thanks to the null space projection technique applied by both approaches.

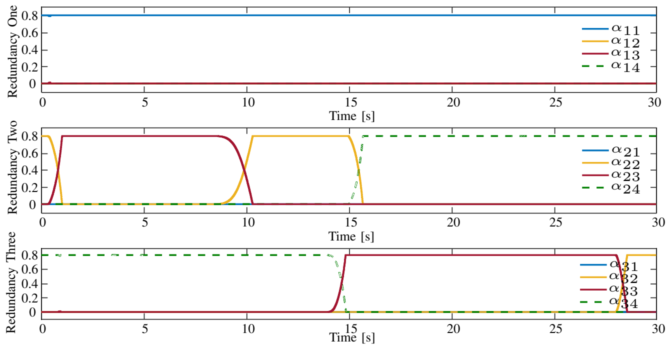
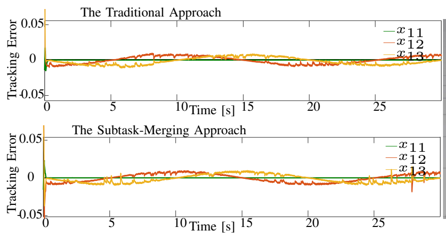


Fig. 7. Three redundancies shift dynamically among four elementary subtasks in due course. The three solid lines correspond to three elementary singularity-avoidances. The dotted line corresponds to the wrist-limit subtask.



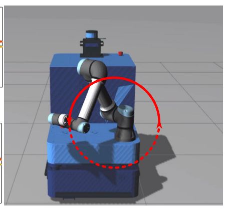


Fig. 8. The manipulator performs a primary task of drawing a circle along a desired end-effector path. Both the traditional and our approach perform well in executing the primary task.

VI. Conclusion and Future Work

This work has addressed the constrained redundancy resolution problems where multiple constraints or subtasks in addition to a primary task have to compete for insufficient redundancies. The proposed approach based on subtask merging and null space projection resolves redundancy insufficiency by dynamically allocates them to subtasks in compliance with task status and priority. Two real robot case studies with solid and substantial results have proved that our approach can be a promising solution to suitably handle complex robot applications characterized by dynamic and unstructured environments. Based on the results, our future works will focus on (1) further modulating and smoothing redundancy shifts to reduce its effect on task execution, e.g. at around 15s in Fig. 6(b), the joint difference fluctuates shortly due to a redundancy shift. and (2) introducing a certain level of predicting capability to the weight updating strategy such as to proactively predict and accommodate the change of task status, e.g. the occurrence of an emergency.

References

[1] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, “Analysis and control of articulated robot arms with redundancy,” *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927–1932, 1981.

[2] B. Siciliano, “Kinematic control of redundant robot manipulators: A tutorial,” *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[3] S. Chiaverini, G. Oriolo, and A. A. Maciejewski, “Redundant robots,” in *Springer Handbook of Robotics*. Springer, 2016, pp. 221–242.

[4] A. A. Maciejewski and C. A. Klein, “Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments,” *The international journal of robotics research*, vol. 4, no. 3, pp. 109–117, 1985.

[5] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, “Task-priority based redundancy control of robot manipulators,” *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[6] F. Ficuciello, L. Villani, and B. Siciliano, “Variable impedance control of redundant manipulators for intuitive human–robot physical interaction,” *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 850–863, 2015.

[7] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[8] S. S. Ge and Y. J. Cui, “Dynamic motion planning for mobile robots using potential field method,” *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[9] F. Flacco and A. De Luca, “Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties,” *Robotics and Autonomous Systems*, vol. 70, pp. 191–201, 2015.

[10] L. Sciavicco and B. Siciliano, “A solution algorithm to the inverse kinematic problem for redundant manipulators,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988.

[11] A. M. Zanchettin and P. Rocco, “A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation,” *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 514–521, 2011.

[12] M. Benzaoui, H. Chekireb, and M. Tadjine, “Redundant robot manipulator control with obstacle avoidance using extended jacobian method,” in *18th Mediterranean Conference on Control and Automation, MED’10*. IEEE, 2010, pp. 371–376.

[13] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, “Dynamic multi-priority control in redundant robotic systems,” *Robotica*, vol. 31, no. 7, p. 1155, 2013.

[14] C. Ott, A. Dietrich, and A. Albu-Schäffer, “Prioritized multi-task compliance control of redundant manipulators,” *Automatica*, vol. 53, pp. 416–423, 2015.

[15] A. Dietrich, C. Ott, and J. Park, “The hierarchical operational space formulation: Stability analysis for the regulation case,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1120–1127, 2018.

[16] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, “Integration of reactive, torque-based self-collision avoidance into a task hierarchy,” *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1278–1293, 2012.

[17] S. B. Slotine, “A general framework for managing multiple tasks in highly redundant robotic systems,” in *proceeding of 5th International Conference on Advanced Robotics*, vol. 2, 1991, pp. 1211–1216.

[18] L. Sentis and O. Khatib, “Synthesis of whole-body behaviors through hierarchical control of behavioral primitives,” *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.

[19] R. Lober, V. Padois, and O. Sigaud, “Variance modulated task prioritization in whole-body control,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3944–3949.

[20] N. Dehio and J. J. Steil, “Dynamically-consistent generalized hierarchical control,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1141–1147.

[21] L. Penco, E. M. Hoffman, V. Modugno, W. Gomes, J.-B. Mouret, and S. Ivaldi, “Learning robust task priorities and gains for control of redundant robots,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2626–2633, 2020.

[22] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, “Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 101–108.

[23] P. Di Lillo, S. Chiaverini, and G. Antonelli, “Handling robot constraints within a set-based multi-task priority inverse kinematics framework,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7477–7483.

VII. Appendix

A. Winner-Take-All Based Updating Algorithm

Algorithm 1 $\dot{\mathbf{A}} = \mathbf{W}(\mathbf{A}, \mathbf{S}, \mathbf{P})$

```
Input: $\mathbf{A}, \mathbf{S}, \mathbf{P}$
Output: $\dot{\mathbf{A}}$
 1: $\dot{\mathbf{A}} = \mathbf{P} \times \mathbf{S}$
 2: for each row $\dot{\mathbf{A}}_i$ in $\dot{\mathbf{A}}$ do
 3:     $\omega = \text{argmax}(\dot{\mathbf{A}}_i)$
 4:     if $\alpha_{i\omega} = \gamma$ then
 5:         $\dot{\mathbf{A}}_i \leftarrow \mathbf{0}$
 6:     else
 7:         $v = \text{argmax}(\dot{\mathbf{A}}_i - \{\dot{\alpha}_{i\omega}\})$
 8:         $z = (\dot{\alpha}_{i\omega} + \dot{\alpha}_{iv})/2$
 9:         $\dot{\mathbf{A}}_i \leftarrow \dot{\mathbf{A}}_i - z$
10:         $s \leftarrow 0$
11:         for $j \leq l$ do
12:             if $\dot{\alpha}_{ij} > 0$ and $\alpha_{ij} \neq 0$ then
13:                 $s \leftarrow s + \dot{\alpha}_{ij}$
14:         $\dot{\alpha}_{i\omega} \leftarrow \dot{\alpha}_{i\omega} - s$
15: return $\dot{\mathbf{A}}$
```

Alg. 1 first yields a preliminary updating rate by multiplying the priority matrix $\mathbf{P}$ with the task status matrix $\mathbf{S}$ (line 1). Then at each row $\dot{\mathbf{A}}_i$, if the entry $\alpha_{i\omega}$ in $\mathbf{A}_i$ corresponding to the greatest update in $\dot{\mathbf{A}}_i$ is already saturated at 1, then $\mathbf{A}_i$ will not be updated by setting $\dot{\mathbf{A}}_i$ to be $\mathbf{0}$ (line 3–5).

Otherwise, the algorithm first lowers $\dot{\mathbf{A}}_i$ to a baseline by subtracting an average of the first-two largest entries (line 7-9). This ensures only one updating rate in $\dot{\mathbf{A}}_i$ is positive, i.e. only one weight $\mathbf{A}_i$ will increase. Then, in order to ensure the sum of the updating rate is 0, we calculate the sum of the current effective updating rate and add it to the maximum update rate (line 10–14).

$$\begin{aligned} S &= \dot{\alpha}_{\omega j} + T \\ T &= \sum_{i \neq \omega \cap (\dot{\alpha}_{ij} > 0 \cup \alpha_{ij} \neq 0)} \dot{\alpha}_{ij} \end{aligned}$$

S of the equation represents the sum of all valid update rates $\dot{\alpha}_{ij}$.

$$\begin{aligned} \dot{\alpha}_{\omega j} &\leftarrow \dot{\alpha}_{\omega j} - S \\ &= \dot{\alpha}_{\omega j} - (\dot{\alpha}_{\omega j} + T) = -T \end{aligned}$$

$$\dot{\alpha}_{\omega j} + \sum_{i \neq \omega \cap (\dot{\alpha}_{ij} > 0 \cup \alpha_{ij} \neq 0)} \dot{\alpha}_{ij} = -T + T = 0$$

The above formula indicates that the sum of all valid update rates is 0. Therefore, after the A matrix is updated, the sum of its items remains unchanged. This will ensure that the weight will not be cleared.

B. Weight Convergence and System Stability

This section presents a detailed proof our approach can converge each weight in A to a stable state along both redundancy and subtask.

Suppose two elementary subtasks f_p and f_q, where f_p is being (or has been) activated, i.e. $\bar{f}_p \simeq 1$, and by contrast f_q is idle, i.e. $\bar{f}_q \simeq 0$. We aim to prove that the weight transition can be always correctly achieved for both subtasks, such that they can be suitably performed in due course. We open up our proof along the redundancy and subtask space separately.

I. Weight Transition along Redundancy: Assume an i-th redundancy is available for subtasking f_p and f_q. If in the winner take all process, the winner is f_p,

$$\Delta\dot{\alpha}_{i-pq} := \dot{\alpha}_{ip} - \dot{\alpha}_{iq} = \mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{ip} - \mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{iq} \geq 0 \tag{19}$$

The weight will transition from f_q to f_p, and vice versa.

If the winner has been born and the maximum update value is still the winner, then the weight of all non-winners is 0, the weight remains stable, and there is no mutual transition (Alg. 1 line 4-5). If there is a weight transitionthe below relationship holds for all i that is not the winner.

$$\frac{\mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{ip} - \mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{iq}}{(\mathbf{PS})_{ip} - (\mathbf{PS})_{iq}} \geq 1 \tag{20}$$

In Alg. 1 lines 7 to 9, only the same item z is subtracted from all elements, and the relative distance between elements remains the same. Since neither f_q nor f_p is winner, there is no action on line 10.

Then the relative updating difference between f_p and f_q is

$$\begin{aligned}\Delta\dot{\alpha}_{i-pq} &:= \dot{\alpha}_{ip} - \dot{\alpha}_{iq}\\ &= \mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{ip} - \mathbf{W}(\mathbf{P},\mathbf{S},\mathbf{A})_{iq} \geq (\mathbf{PS})_{ip} - (\mathbf{PS})_{iq}\\ &= \prod_{u=0}^{i-1}(1-\alpha_{up})\prod_{v=0}^{p-1}(1-\alpha_{iv})\prod_{u\neq i}(\gamma-\alpha_{up})\bar{f}_p\\ &- \prod_{u=0}^{i-1}(1-\alpha_{uq})\prod_{v=0}^{q-1}(1-\alpha_{iv})\prod_{u\neq i}(\gamma-\alpha_{uq})\bar{f}_q\end{aligned}$$

Specifically, there are four cases:

Case One: Suppose neither of f_p,f_q is occupying a redundancy, i.e. $\alpha_{up} \simeq \alpha_{uq} \simeq 0$, $\forall u \neq i$. Then we have

$$\begin{aligned}0 &< \prod_{u=0}^{i-1}(1-\alpha_{up})\prod_{u\neq i}(\gamma-\alpha_{up})\\ &\simeq \prod_{u=0}^{i-1}(1-\alpha_{uq})\prod_{u\neq i}(\gamma-\alpha_{uq}) \simeq \gamma^{n-m-1}\end{aligned}$$

Denote $c = \gamma^{n-m-1} > 0$ (a constant), then we have

$$\dot{\alpha}_{i-pq} \geq (\mathbf{PS})_{ip} - (\mathbf{PS})_{iq} \simeq c(\prod_{v=0}^{p-1}(1-\alpha_{iv})\bar{f}_p - \prod_{v=0}^{q-1}(1-\alpha_{iv})\bar{f}_q)$$

(1). If $p < q$, we have

$$\dot{\alpha}_{i-pq} \geq c\prod_{v=0}^{p-1}(1-\alpha_{iv})(\bar{f}_p - \prod_{v=p}^{q-1}(1-\alpha_{iv})\bar{f}_q)$$

which indicates as $\bar{f}_p$ approaches one and $\bar{f}_q$ approaches zero in line with their task status, $\dot{\alpha}_{i-pq} \geq 0$ is guaranteed, i.e the weight of f_p will increase relatively faster and therefore a higher task priority is correctly given to f_p.

(2). If $p > q$, similarly, we have

$$\dot{\alpha}_{i-pq} \geq c\prod_{v=0}^{q-1}(1-\alpha_{iv})(\prod_{v=q}^{p-1}(1-\alpha_{iv})\bar{f}_p - \bar{f}_q)$$

which indicates, similarly, a higher weight will be eventually transited to f_p, as $\bar{f}_p$ and $\bar{f}_q$ vary in accordance with their task status. It also suggests that, however, since f_q is previous to f_p by index, until $\bar{f}_q = 0$, $\dot{\alpha}_{i-pq} \geq 0$ is not guaranteed. That is, the weight of f_p will not be improved as faster as f_q until f_q is competed, since f_q has a higher indexing priority.

Case Two: Suppose only f_p is occupying a redundancy, i.e. $\exists u \neq i, \alpha_{up} = \gamma$. Then $\dot{\alpha}_{ip} = 0$ and therefore $\dot{\alpha}_{i-pq} = \dot{\alpha}_{ip} - \dot{\alpha}_{iq} = 0 - \dot{\alpha}_{nq} \leq 0$. That is, a relatively faster weight increase will be given to f_q. This is in compliance with the fact that $\bar{f}_p$ has been allocated with a redundancy and therefore its weight will not increase. A higher weight will be accordingly transited to f_q.

Case Three: Suppose only f_q is occupying a redundancy, similarly, we can prove $\dot{\alpha}_{i-pq} \geq 0$ holds, which is consistent with the fact a higher weight is supposed to transit to f_p.

Case Four: Suppose both substasks are holding redundancies. Then $\dot{\alpha}_{ip} = \dot{\alpha}_{iq} = 0$ and therefore $\dot{\alpha}_{i-pq} = 0$, i.e. there is no relative difference between their updating rate, which is consistent with the fact that subtasks that have been (being) executed will not compete for redundancy and there is no weight transition between them.

II. Weight Transition along Subtask: Suppose the subtask f_p has been allocated to a u-th redundancy, i.e. $\exists u, \alpha_{up} = \gamma$. Then at any other ω-th redundancy, it satisfies $\dot{\alpha}_{\omega p} \leq 0, \forall \omega \neq u$. That is, once a subtask has been allocated at a certain redundancy, the weights of the subtask at other redundancies will not increase, which exactly meets the constraint that an assigned subtask should not jump back and forth.

To sum up, our approach can converge the weights along both the redundancy and the subtask space. Since each subtask controller is stable in design, the entire system can be executed stably once the convergence is achieved.