Takeout Service Automation With Trained Robots in the Pandemic-Transformed Catering Business

Ting-Yu Lin, *Member, IEEE*, Kun-Ru Wu, You-Shuo Chen, Wei-Hau Huang, and Yi-Tuo Chen

***Abstract*—Under strict social-distancing directives during the COVID-19 pandemic, one of the most impacted businesses is probably the catering industry, which has been forced to earn their revenue mainly from delivery and takeout services. However, traditional takeouts require patrons to wait in line, order and pick up their meals, incurring unnecessary human contact and service inefficiency. We observe these drawbacks and propose a contactless meal order and takeout service (Mots) automated system realized by AI-assisted smart robots to address the issue. In our Mots system, we develop a bump-free schedule based on the Welsh-Powell coloring algorithm for grouping robots into several non-colliding moving batches. Simulation results show that our Mots solution can effectively improve takeout efficiency and promote service accuracy, boosting business profits up to 95.4% under simulated cases for various cafeteria scales and shop popularity differences, compared to the traditional takeout method. Our experiments suggest that Mots is also capable of accommodating a sudden surge of arriving patrons within a short period of time. Furthermore, we have implemented a proof-of-concept prototype to demonstrate our Mots automated operations.**

***Index Terms*—AI-enabled robotics, service robotics, embedded systems for robotic and automation, automation technologies for smart cities, collision avoidance.**

I. Introduction

As of mid January 2021, approximately 91 million people have been infected with the coronavirus and 1.95 million deaths have been reported worldwide. Now the daily estimate of COVID-19 confirmed cases still remains high. Our world has been experiencing a vast lifestyle change due to the global pandemic. Social-distancing directives for combating the ongoing pandemic have greatly impacted the catering industry, forcing restaurants to do their business mainly from delivery and takeout services. Although the deliver-to-your-home service is convenient, it often comes with extra delivery fees, so many people prefer picking up takeout meals themselves.

We observe that, in a busy mall environment or campus cafeteria during lunch time, depicted in Fig. 1(a), traditional waiting lines in front of food shops typically vary in length due to different popularity. For example, among the first eight arriving patrons, six choose Burger, generating the longest line, which limits overall system efficiency. Moreover, regular takeout process involves close physical contact, highly discouraged during the pandemic. To avoid unnecessary human contact and increase takeout efficiency, we propose an automated robotic system, displayed in Fig. 1(b). Our contactless meal order and takeout service (Mots) offers online orders so that meals can be prepared beforehand, and allows patrons to arrive at any empty spot, without having to wait in a long line. Those waiting spots can be designed in a semicircular shape at fixed locations, easier to maintain safe social distances between patrons in a space-limited mall/cafeteria. Arriving patrons fill in the spots starting from row #1, then row #2, and so on. This way helps keep the waiting lines short. For the first eight[1] arriving patrons, our system will be notified and their meals will be carried by smart robots initially located at arbitrary departure positions. Once robots get ready to serve row #1 patrons, a bump-free scheduling algorithm comes into play.

As illustrated in Fig. 1(b), potential bumps between robots should be identified before dispatching meals to patrons. Then a bump-free schedule divides robots into moving groups that do not collide. The bump-free grouping strategy is non-trivial as it can be reduced from the known NP coloring problem, meaning that currently there does not exist any effective approach able to produce an optimal coloring (grouping) solution in polynomial time. To provide a pragmatic solution, our algorithm has been judiciously designed to generate perhaps suboptimal, yet efficient enough robots moving schedules. Section II will elaborate on the algorithmic details of our bump-free scheduling. Fig. 1(c), (d) demonstrates one such possible schedule with two moving batches, where Batch #1 robots ($R_1, R_3, R_4, R_6, R_7, R_8$) move simultaneously and then Batch #2 (R_2, R_5) follows to complete the process. Now, our contactless Mots system has successfully dispatched meals to row #1 patrons with minimal human contact by maintaining social distances at all times, while effectively

Manuscript received October 14, 2020; accepted January 7, 2021. Date of publication January 18, 2021; date of current version February 4, 2021. This letter was recommended for publication by Associate Editor E. Lanzarone and Editor J. Yi upon evaluation of the reviewers' comments. This work was co-sponsored by ITRI, Gunitech Corp., Pervasive Artificial Intelligence Research (PAIR) Labs, Center for Open Intelligent Connectivity (Featured Areas Research Center Program within the framework of Higher Education Sprout Project), and the Ministry of Science and Technology (MOST) of Taiwan under Grants #109-2634-F-009-026 and #105-2218-E-009-003. *(Corresponding author: Ting-Yu Lin.)*

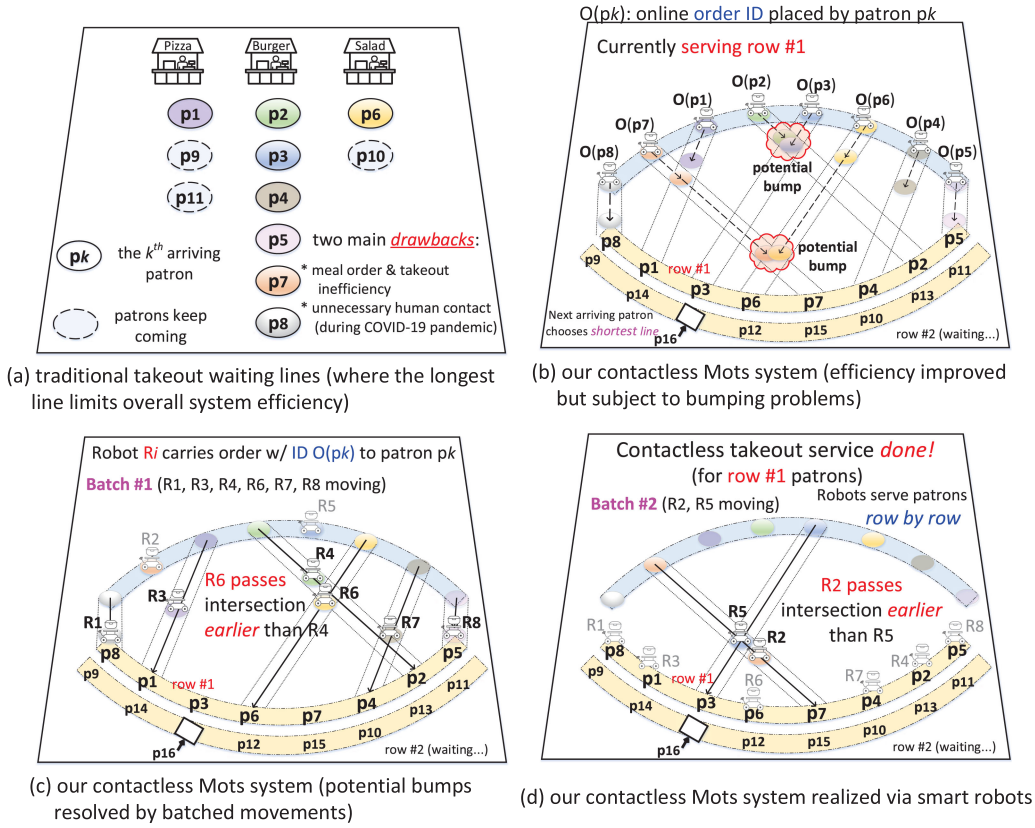
Ting-Yu Lin, You-Shuo Chen, Wei-Hau Huang, and Yi-Tuo Chen are with the College of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu City 30010, Taiwan (e-mail: tingyoyo@nctu.edu.tw; std.91182@gmail.com; greensky8877@gmail.com; cydonnny@gmail.com).

Kun-Ru Wu is with the Department of Computer Science, National Chiao Tung University, Hsinchu City 30010, Taiwan (e-mail: wufish@g2.nctu.edu.tw).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3052451, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3052451

[1]This number only refers to our Fig. 1 example. In typical catering scenarios, depending on available space (cafeteria scales) and usable robots, a much larger number can be applied to encourage higher degree of parallelism in serving patrons during peak hours.



(a) traditional takeout waiting lines (where the longest line limits overall system efficiency)

(b) our contactless Mots system (efficiency improved but subject to bumping problems)

(c) our contactless Mots system (potential bumps resolved by batched movements)

(d) our contactless Mots system realized via smart robots

Fig. 1. Comparison of the traditional takeout waiting lines and our contactless meal order and takeout service (Mots).

improving overall takeout efficiency by practicing parallelism to speed up the process. We further investigate the system performance issue (service efficiency and business profits) in Section III.

Although the life-threatening crisis has brought us various challenges, we believe it also represents an opportunity to accelerate the automation industry. Our goal is to contribute a practical IoT-based contactless meal takeout solution via AI-assisted smart robots during the pandemic to minimize physical human contact while maintaining satisfactory dietary needs. We propose to equip each robot with AI-enabled image recognition function to make the whole takeout experience truly automated, with a sense of service accuracy and promptness. Our Mots system architecture and operation flow are described in Section IV. Furthermore, Section V reports a real-life Mots prototype with demo video available at [1].

A. *Featured Benefits*

We summarize our Mots features as follows.

- Compared to traditional meal takeout service, which involves unnecessary human touch, our *contactless* system enables patrons to *maintain safe social distances* throughout the whole process.
- Our Mots offers online orders and serves patrons *efficiently*. For restaurants that also allow orders in advance [2], arriving patrons still need to wait in line for pick-up, served sequentially. By deploying smart robots to serve patrons row by row, Mots encourages service parallelism and effectively speeds up the takeout process, providing *service promptness*.
- With readily-available image recognition technology [3], [4], we propose to incorporate AI-assisted smart robots in our Mots system to reduce human errors, which often occur when manually retrieving ordered meals for arriving patrons. By eliminating unnecessary, yet sometimes inevitable, human flaws during busy meal pick-up hours, Mots can further improve takeout *service accuracy*.

II. Contactless Meal Order and Takeout Service (Mots) via Smart Robots

To realize a practical solution, our system allows shop owners to place a patron's ordered meal on any available robot R_i (preferably closest to the shop's location) and then Mots server will take care of meal dispatch to the correct patron. Envision a busy mall/cafeteria environment, being able to dispatch quickly (and correctly) is essential. In Fig. 1(b), prepared meals are carried by arbitrary robots to row #1 patrons. Since both the departing robots and arriving patrons are in no particular order, we need mechanisms to predict potential bumps before performing dispatch. Based on basic geometry calculations, we design one such mechanism for bumping prediction in Section II-A. Once the bumping relationship between robots is identified, we introduce our bump-free scheduling algorithm in Section II-B, and suggest potential improvements on overall service time in Section II-C and Section II-D, respectively.

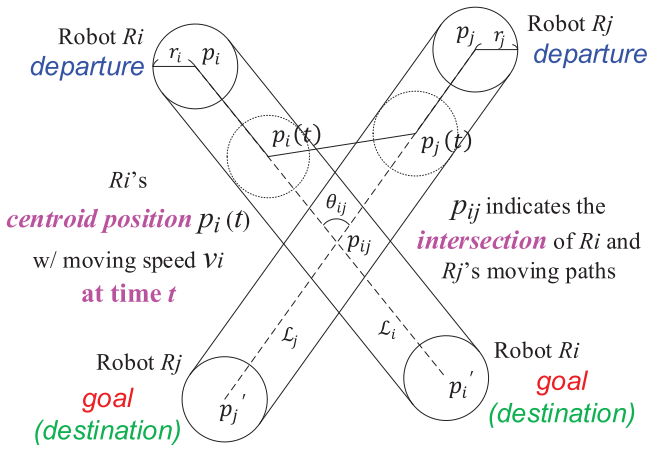


Fig. 2. Potential bumping prediction between any two moving smart robots R_i and R_j before meal dispatch takes place.

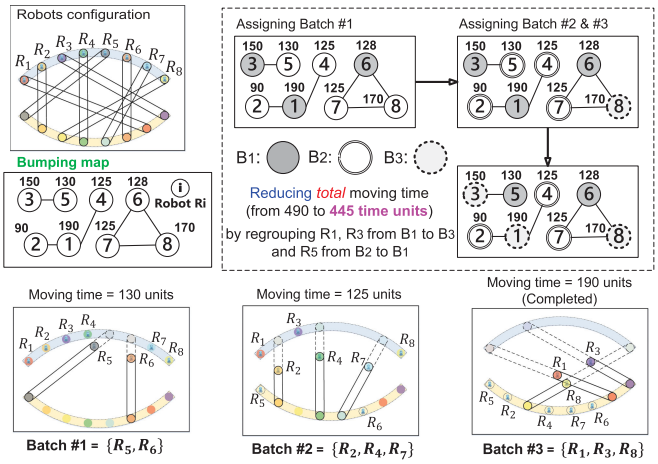


Fig. 3. Bump-free scheduling example in our Mots system.

A. Robots Bumping Prediction

In a previous work [5] aiming to schedule collision-free moving paths for mobile sensors, each sensor is assumed to be a tiny point (with negligible volume) on a 2-D plane. However, in realistic settings, moving robots occupy non-negligible space. Hence we propose to model each robot R_i as a circle centered at point p_i with radius r_i, heading for destination point p_i' at moving velocity v_i. The travel paths for R_i and R_j can be formulated as lines L_i and L_j respectively, as shown in Fig. 2, where p_{ij} indicates the *intersection* of R_i and R_j's travel paths (which can be obtained by solving the two line equations) and θ_{ij} represents the angle generated by the two lines. Define $p_i(t)$ as robot R_i's centroid position at time t. According to the **Cosine Theorem**, we have $\cos\theta_{ij} = (\overline{p_ip_{ij}}^2 + \overline{p_jp_{ij}}^2 - \overline{p_ip_j}^2)/(2 * \overline{p_ip_{ij}} * \overline{p_jp_{ij}})$. Consequently, the distance $\overline{p_ip_j}(t)$ between robots R_i and R_j at time t can be obtained by first deriving $\overline{p_ip_j}^2(t)$ as follows.

$$\begin{aligned}\overline{p_ip_j}^2(t) &= \overline{p_i(t)p_{ij}}^2 + \overline{p_j(t)p_{ij}}^2 - 2 * \overline{p_i(t)p_{ij}} * \overline{p_j(t)p_{ij}} * \cos\theta_{ij} \\ &= (\overline{p_ip_{ij}} - v_it)^2 + (\overline{p_jp_{ij}} - v_jt)^2 \\ &\quad - 2(\overline{p_ip_{ij}} - v_it)(\overline{p_jp_{ij}} - v_jt)\cos\theta_{ij} \\ &= (v_i^2 + v_j^2 - 2v_iv_j\cos\theta_{ij})t^2 \\ &\quad + 2((v_i\overline{p_jp_{ij}} + v_j\overline{p_ip_{ij}})\cos\theta_{ij} - v_i\overline{p_ip_{ij}} - v_j\overline{p_jp_{ij}})t \\ &\quad + \overline{p_ip_{ij}}^2 + \overline{p_jp_{ij}}^2 - 2 * \overline{p_ip_{ij}} * \overline{p_jp_{ij}} * \cos\theta_{ij}\end{aligned}$$

Let

$$\begin{aligned}a &= (v_i^2 + v_j^2 - 2v_iv_j\cos\theta_{ij}) \\ b &= 2((v_i\overline{p_jp_{ij}} + v_j\overline{p_ip_{ij}})\cos\theta_{ij} - v_i\overline{p_ip_{ij}} - v_j\overline{p_jp_{ij}}) \\ c &= \overline{p_ip_{ij}}^2 + \overline{p_jp_{ij}}^2 - 2 * \overline{p_ip_{ij}} * \overline{p_jp_{ij}} * \cos\theta_{ij}\end{aligned}$$

We have $\overline{p_ip_j}^2(t) = at^2 + bt + c$. To ensure robots R_i and R_j do not bump into each other, the distance $\overline{p_ip_j}(t)$ must remain larger than the summation of R_i and R_j's radius $r_i + r_j$ at all times, that is $\overline{p_ip_j}(t) > r_i + r_j$ for any time t. As distance is always non-negative, by squaring both sides of the inequality, we have $\overline{p_ip_j}^2(t) > (r_i + r_j)^2$. Define function $f(t) = \overline{p_ip_j}^2(t) - (r_i + r_j)^2 = at^2 + bt + c - (r_i + r_j)^2$. Suppose $T_{travel} = \min\{moving\_time_i, moving\_time_j\}$ indicates the shorter travel time for robots R_i and R_j to complete its dispatch. Since $f(t)$ is a function of time t, as long as the $f(t)$ curve does not intersect (come across) with the t-axis during time interval $[0, T_{travel}]$, we can infer robots R_i and R_j are bump-free. Now, the bump-free criterion has been derived, based on which we can allow two robots to travel simultaneously (in the same moving batch) if they satisfy the criterion; otherwise, they must be scheduled in different moving batches. In our system, robots are equipped with collision-detection sensors to prevent occasional bumps in case of prediction inaccuracy. When that happens, involved robots will stop and wait for Mots server's instructions before proceeding further.

B. Bump-Free Scheduling Algorithm

After performing robots bumping prediction, we obtain the *bumping map* by connecting any two robots with an edge if they violate the bump-free criterion, depicted in Fig. 3, to reflect the *conflict* relationship of robots' moving paths. Given the bumping map, we develop our bump-free scheduling algorithm as follows. Perceive scheduled moving batches as different applied colors. Consider an undirected graph, assigning bump-free batches is equivalent to vertex-coloring the graph such that any two connected vertices are applied different colors. Recall the *k-coloring* NP problem in graph theory, which can be formally reduced to our batch assigning problem.[2] Similarly, obtaining a bump-free schedule with the *optimal* (fewest) k batches is not always possible within a reasonable computation time. Hence, we need to compromise by using existing polynomial-time coloring schemes, such as the Welsh-Powell algorithm [6]–[8], to produce a perhaps suboptimal, yet efficient enough, bump-free schedule.

[2]Due to space limitation and to avoid using complex mathematical terms, we omit the proof details in the article.

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
S	Set of smart robots currently in service
$bumping[i][j]$	Boolean flag to indicate whether or not robots R_i and R_j present a potential bump at the intersection
B_i	Robot R_i's bumping set (conflict robots with R_i)
$valence_i$	Robot R_i's vertex degree in bumping map
$moving\_time_i$	Robot R_i's travel time from departure to goal
S_{order}	Set of robots ordered in descending valences
$color_i$	Robot R_i's color number (value)
c_{max}	Used colors count (total number of used colors)
U	Set of used colors (collection of color values)

We start coloring (assigning a moving batch) from the vertex (robot) having the highest valence. Here we adopt the term *valence*, as specified in the Welsh-Powell algorithm, to indicate the *vertex degree*, which means the number of connected edges involved with a vertex. A valence value reflects the vertex's *degree of conflict* in the bumping map. Define an ordered set S_{order} to include all robots in order of descending valences. In Fig. 3, for example, R_1, R_6, R_7, and R_8 all have the highest valence 2, we propose to break ties using robot's ID with smallest ID considered first. So R_1 will be assigned to the first moving batch B_1 and then all other non-conflict robots (examined sequentially in S_{order}) including R_6 and R_3 can be assigned in B_1 as well. Then the assigning (coloring) procedure moves on to consider next unassigned (uncolored) robot in set S_{order}, assigning R_7 to batch B_2, and all other non-conflict robots including R_2, R_4, and R_5 can also be assigned in B_2. Finally, R_8 is assigned to B_3 and bump-free schedule completes. Our idea is to take care of robots with largest degree of conflict as early as possible, so that the *number* of required *moving batches* can hopefully be *reduced or minimized*.[3] Table I summarizes used notations and Algorithm 1 provides our bump-free scheduling pseudocode.

C. Enhanced Overall Service Time

Based on Algorithm 1, we further improve on the overall service time by considering robots' moving times. Take Fig. 3 for instance, the number besides each robot indicates its moving time (in time units). During the bump-free scheduling process, our algorithm keeps track of *dominant* robot (with *maximum moving time*) in every assigned batch. In our example, R_1 is the dominant robot (with dominant moving time 190 in B_1), whereas R_5 and R_8 are dominant robots (with dominant moving times 130 and 170) in B_2 and B_3 respectively. We observe that, by switching R_1 from B_1 to B_3, we can effectively *decrease* B_1's moving time by 40 time units (from 190 to 150) but *increase* B_3's moving time by 20 time units (from 170 to 190), resulting in a *net profit* of decreased 20 time units. Similarly, by shifting R_3 from B_1 to B_3, a *net profit* of decreased 22 time units can be benefited. Furthermore, by shifting R_5 from B_2 to B_1, we obtain a *net profit* of decreased 3 time units. Consequently, the overall service time T_{total} has been shortened (from 490 to 445 time units) due to the above batch switching actions.

[3]Since the batch assigning problem is NP complete, we cannot guarantee the total batch number obtained from our bump-free scheduling algorithm to be minimum (optimal).

Algorithm 1: Mots Bump-Free Scheduling.

include all k smart robots in set S;
initiate $bumping[i][j] = 0\ \forall R_i, R_j \in S$; // $i = 1, \ldots, k$
for each robot pair (R_i, R_j) where $i \neq j$ **do**
  if R_i and R_j predicted to bump into each other **then**
    $bumping[i][j] = 1$; // with moving speeds v_i and v_j
end for
clear $valence_i = 0$; // vertex degree for R_i
calculate $moving\_time_i\ \forall R_i \in S$;
for (i=1, i++, $i \leq k$) **do**
  initiate bumping set $B_i = \emptyset$;
  for ($j = i + 1$, j++, $j \leq k$) **do**
    if $bumping[i][j] = 1 \parallel bumping[j][i] = 1$ **then**
      include R_i into bumping set B_j;
      include R_j into bumping set B_i;
      $valence_i + +$; $valence_j + +$;
  end for
end for
// bumping map successfully established
sort the coloring order $\forall R_i \in S$ in descending valences;
//R_i with max $valence_i$ colored first
establish ordered set S_{order} to replace S;
// index differs in two sets; for R_i, R_j with same valence, R_i
// will be considered before R_j if $i < j$
clear $color_i = 0$, $c_{\max} = 0$; // used colors count
initiate used colors set $U = \emptyset$;
initiate temporary bumping set $B = \emptyset$;
initiate set $dominant_{robot}[i] = \emptyset$; $dominant_{time}[i] = 0$;
for (i=1, i++, $i \leq k$) **do**
  if $color_i \neq 0$ **then continue**;
  $c_{\max} + +$;
  $U = U \cup c_{\max}$; $color_i = c_{\max}$;
  $dominant_{robot}[color_i] = R_i$;
  $dominant_{time}[color_i] = moving\_time_i$;
  temporary bumping set $B = B_i$;
  for ($j = i + 1$, j++, $j \leq k$) **do**
    if $color_j \neq 0$ **then continue**;
    if $R_j \notin B$ **then**
      $color_j = color_i$;
      $B = B \cup B_j$;
      if $moving\_time_j > dominant_{time}[color_i]$ **then**
        $dominant_{robot}[color_i] = R_j$;
        $dominant_{time}[color_i] = moving\_time_j$;
  end for
end for // bump-free scheduling (coloring) completed
clear $T_{total} = T_{B_i} = 0$; // moving time for batch i
for (i=1, i++, $i \leq c_{\max}$) **do**
  $T_{total} = T_{total} + T_{B_i}$; // overall service time
end for

By going through every color (assigned batch) in used colors set U, we examine the corresponding dominant robot and seek any other possible batch re-arrangements with reduced T_{total}. In this way, we accelerate the meal dispatching process, further enhancing our *service promptness*.

D. Flexible Grouping Strategy

As an advanced function, we can also adjust robot R_i's travel velocity v_i to reduce bumping cases. By slowing down or accelerating velocities, some potential bumps can be resolved, possibly leading to fewer required moving batches. Furthermore, for patrons ordering the same meal contents, Mots server can be designed to flexibly alternate moving paths, selecting a schedule with smaller T_{total}. Ideally, adopting *flexible* grouping strategies enables our system to potentially achieve perfect parallelism by sending out only one batch to serve a single row of patrons. This part of algorithmic details requires more sophisticated investigation, and will be directed into our future work.

E. Computation Complexity

Suppose there are n robots serving in a row. Our bump-free scheduling algorithm first involves solving pair-wise line equations and examining corresponding function $f(t)$, which requires O(n^2) time complexity. Then the vertex-coloring process for batches assignment adds the computation time by O(n^2). Assume k colors have been applied, where $k \leq n$, we can easily infer the maximum number of robots assigned in a batch is $n - k + 1$. For possible enhancement on overall service time, the batches switching action examining through robots in k batches results in additional time O(nk). The above performed computations thus render our algorithm's time complexity in asymptotic upper bound O(n^2). To further reduce the complexity, we set i robots (currently $i = 8$) as a computation group in our program, which requires constant time O(i^2). For n robots, there will be $\lceil n/i \rceil$ computation groups, resulting in O(ni) computation time. This way may not produce optimal solutions but can effectively reduce our algorithm's complexity in linear time O(n).

III. Analysis on Takeout Service Efficiency and Business Profits

In mall environments, typically the service hours are fixed. Without automation, a conventional shop can only serve a certain number of patrons in a day. Our Mots system expects to promote business sales by providing service efficiency, which not only attracts more patrons coming, but also brings in better business income by successfully having more meals sold during open hours. Suppose patrons keep coming during open hours, the automated Mots system is capable of accommodating a higher number of patrons, hence stimulating greater business profits, compared with the traditional, slow takeout method. From this perspective, service efficiency can be translated into potential business profits.

We first provide a comprehensible analysis on Mots performance by studying the example in Fig. 1. Assume the catering business makes μ profits (e.g., dollars) from each patron per unit time (e.g., per minute). Suppose serving a patron takes one time unit t. In traditional takeout scenarios, as the longest waiting line contains six patrons in Fig. 1(a), the system takes $6t$ in serving the first eight arriving patrons, producing system profits of $8/6 * \mu = 1.33\,\mu$ per unit time. On the other hand, suppose a single batch of robots dispatch takes one time unit, by sending out two batches of smart robots in Fig. 1(c) and (d), our Mots system achieves parallelism and spends $2t$ in serving the eight row #1 patrons, yielding profits of $8/2 * \mu = 4\,\mu$ per unit time, more than *three times higher* business profits compared to the traditional takeout model.

Suppose there are f food shops creating uneven waiting lines with lengths W_i, $i = 1, \ldots, f$, we have the longest waiting length $W_{\max} = \max\{W_i\}$ in conventional scenarios. Assume the time for taking order and preparing meal for one patron is $T_{\text{order}} + T_{prep}$, then *service efficiency* for traditional takeouts can be expressed by $\frac{1}{W_{\max} * (T_{\text{order}} + T_{prep})}$. If a meal order and preparation can be done in one minute, traditional service model in Fig. 1(a) requires approximately six minutes to serve the first eight patrons, where system performance is limited by the longest (most popular) waiting line. The *business profits* (per minute) can be expressed as $\#patrons \times \frac{\mu}{W_{\max} * (T_{\text{order}} + T_{prep})}$.

To relieve the service bottleneck, our Mots solution enables smart robots to serve patrons simultaneously (in parallel) in a single batch. As all meals have been ordered online and prepared beforehand (making $T_{\text{order}} + T_{prep}$ negligible), service time depends on the number of moving batches computed by our bump-free scheduling algorithm. Suppose T_{B_i} denotes the moving time for batch B_i, $i = 1, \ldots, c_{\max}$, where $c_{\max}$ is the used colors count (recall Table I and Algorithm 1). Then *service efficiency* for our Mots system can be expressed by $\frac{1}{\sum_i T_{B_i}}$, and we have *business profits* (per minute) equal to $\#patrons \times \frac{\mu}{\sum_i T_{B_i}}$. In Fig. 3 with three bump-free batches scheduled, the total service time $T_{total} = \sum_{i=1}^{3} T_{B_i} = 490$ or 445 (with batch-switching enhancement proposed in Section II-C) time units. Here we define one unit time as the average time our smart robot takes to travel (move) one grid in distance. From real measurements in our Mots prototype system, a time unit approximates 0.1 seconds. Therefore, in this example, Mots is capable of serving the first eight patrons in less than one minute (49 or 45 seconds), achieving more than *six times better* service efficiency compared to the traditional case.

In realistic settings, by expanding the cafeteria area in Fig. 3 approximately four times in size, Fig. 4 shows the earned profits (times μ per minute) under different patrons arrival rates and shop popularity variances. Assuming three shops with the most popular one attracting 1/2, 2/3, and 3/4 arriving patrons, we simulate small, medium, and high popularity variances among shops. From small-scale to large-scale cafeterias, we generate 8 (mildly busy) up to 128 (extremely busy) patrons arrival per minute. The number of patron arrivals is deterministic, yet with arbitrarily assigned positions. We simulate randomly positioned patrons and apply our bump-free scheduling algorithm to obtain the profits produced by Mots. As patrons arrival rate increases, traditional takeout service makes relatively fixed (limited) amount of business income under a certain popularity distribution, and the earned profits decrease (dropped from 2μ to $1.5\,\mu$ then to $1.33\,\mu$) when popularity variance rises. In contrast, our Mots model enables service parallelism by dispatching meals in batches, and thus encourages remarkable profits boost, especially when shop popularity variance increases, as shown

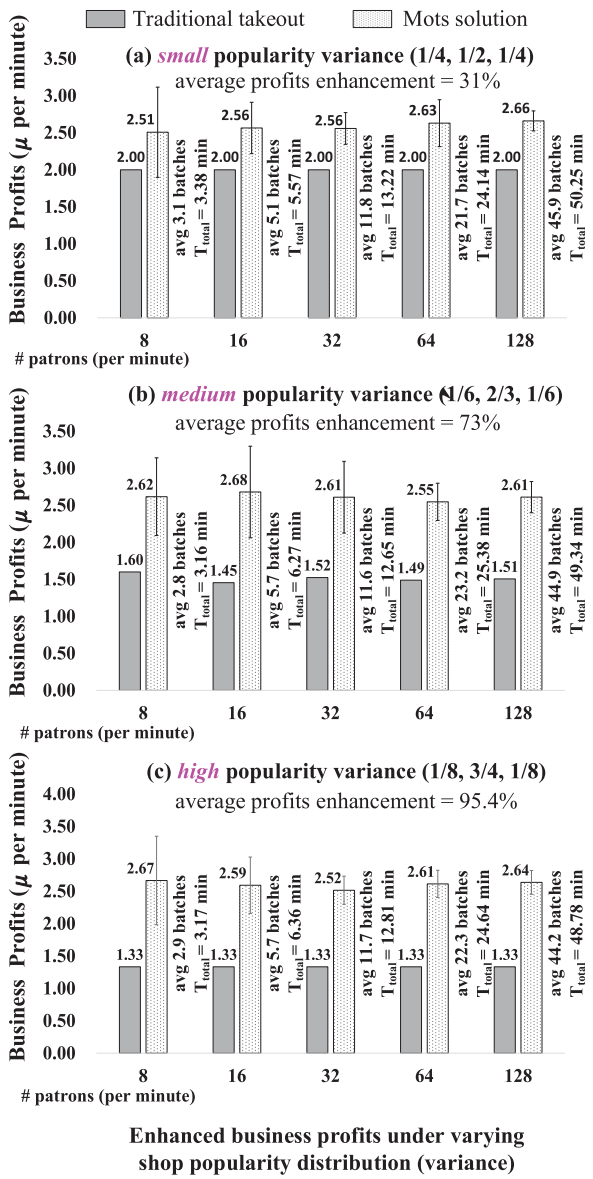


Fig. 4. Potential business profits generated per unit time comparing traditional takeout model and our Mots solution (assuming three food shops) under different patrons arrival rates and shop popularity variances.

in Fig. 4(c). For the Mots simulations, each data point has been obtained from an average of 15 experiments. Under all simulated cases (different cafeteria scales and shop popularity variances), our Mots solution promotes an average of improved 66.5% (up to 95.4%) higher business profits than the traditional takeout model. These results suggest that Mots is not only efficient but also capable of accommodating a sudden surge of arriving patrons within a short period of time. We additionally provide the actual computation time in Table II for all simulated cases, with statistics obtained from our Mots server, which is a general-purpose desktop computer (Intel i5-8400 2.8 GHz CPU, 32 GB RAM, Ubuntu 20.04 LTS). The results show that the required computing time relates to the number of patrons while irrelevant to shop popularity variance. All computation times are in milliseconds. Table II verifies our bump-free algorithm's

TABLE II
AVERAGE MOTS COMPUTATION TIME (MS)

# patrons (per minute)		8	16	32	64	128
shop popularity variance	small	0.31	0.55	1.22	2.27	4.78
	medium	0.31	0.57	1.17	2.28	4.72
	high	0.29	0.56	1.12	2.28	4.61

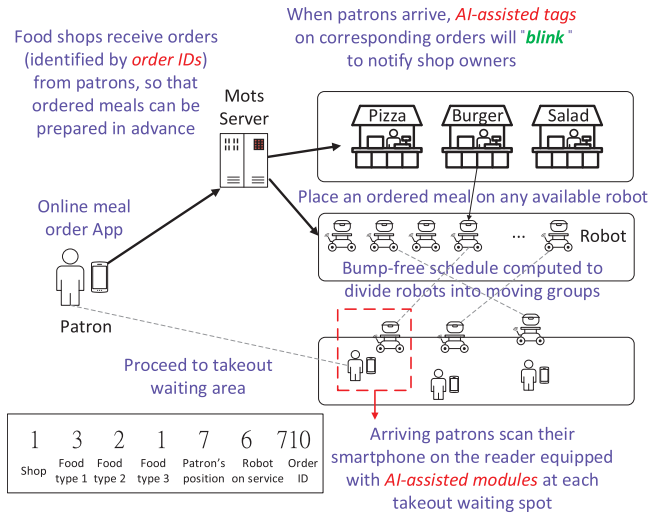


Fig. 5. Mots operation flow and system architecture.

computation time is substantially negligible compared to robots moving time.

From this set of experiments, we corroborate the service efficiency and potential business profits brought by Mots regardless of varying cafeteria scales and shop popularity distributions. As service efficiency enhances to bring in pronounced, noticeable business profits, we believe the concepts and technologies proposed in our Mots solution can attract more company investments in catering automation.

IV. MOTS SYSTEM ARCHITECTURE

Fig. 5 depicts our Mots operation flow and system architecture. Currently we set up a Linux-based (Ubuntu 16.04) Mots server, use Android smartphones, and adopt TurtleBot3 (model Burger) trained as smart robots in our system. Apps are coded in JAVA (smartphones) and Python (server, robots). We have developed a proof-of-concept prototype, with which we introduce AI image recognition technology [3], [4] to cooperate. By attaching an AI-assisted tag to every ordered meal, food shop owners can be spared from manually notifying Mots server and, similarly, Mots server can communicate directly with tags without going through shop owners. Furthermore, arriving patrons can take advantage of AI-assisted modules equipped at takeout waiting spots. Once a patron has been physically present at a takeout waiting spot, he/she scans the QR code on the camera reader to automatically notify Mots server. Then a patron's order ID will be linked with the corresponding meal tag, instructed by Mots server to start "blinking". The blinking action conveniently, visually, reminds shop owners to place the meal on any available

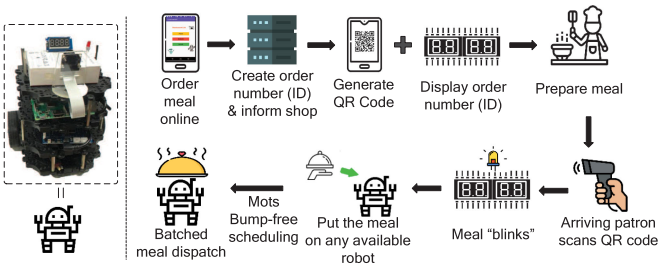


Fig. 6. ROS-based TurtleBot3 and Mots system flow.

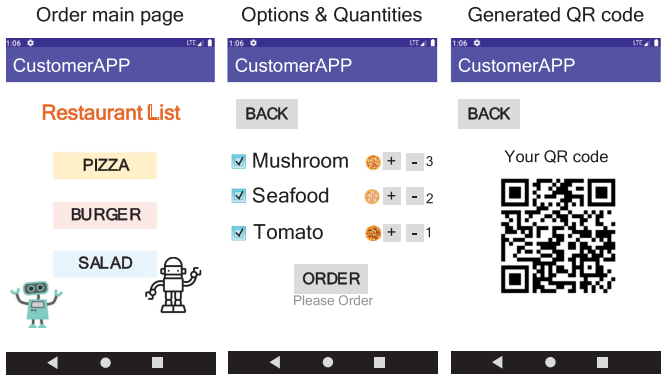


Fig. 7. Online order App user interface (UI).

robot. This feature additionally spares shop owners from the hassle of checking app notifications from Mots server. Furthermore, through AI image recognition technology, our robots can quickly communicate with Mots server and confirm the correctness of meal contents. Those AI-enabled automated functions are not only convenient, but also helpful in eliminating human errors, especially when dealing with a large number of incoming orders during peak hours, promoting *service accuracy*. Apparently, service accuracy is an attractive feature to companies attempting to provide both quick and quality (precise) meal order and takeout experience, with minimal human mistakes.

V. Prototyping Experiences

In this section, we share our prototyping experiences. Our current Mots prototype employs eight smart robots that can be activated to serve arriving patrons in a row. Specific implemented functionalities are described as follows.

A. Robot Hardware and System Flow

Fig. 6 illustrates an ROS-based TurtleBot3 (Burger Pi) [9] and detailed Mots system flow. Every smart robot is equipped with Raspberry Pi 3 embedded board [10], OpenCR [11], Pi camera [12], plus 4-digit seven-segment display module (TM1637) [13] and LED. When an ordered meal has been placed on the smart robot, the corresponding order ID shown on the display is automatically recognized by AI-assisted camera module and, along with robot ID, brought back to Mots server. Then the robot awaits moving instructions from Mots server, which performs careful bump-free scheduling calculations to serve arriving patrons in batches. From our prototyping experiences, TurtleBot3 Burger sometimes brings itself to a short pause in the midst of a traveling path. One reason is possibly due to the uneven floor on which our robots travel, leading to some mechanical problem. We have been trying to investigate other reasons of the sudden halt and work on the moving smoothness in our experiments. But overall, TurtleBot3 delivers a pretty good job as a trained smart robot.

B. Online Order App and Backend Server

In our prototype, online order App has been coded in JAVA on Android platform. Fig. 7 displays the user interface (UI) of the App pages. Every placed order is finished with a successfully generated QR code, containing meal contents identified by an order ID. By simply scanning the QR code at the takeout waiting spot, an arriving patron automatically informs Mots server of both the order ID and patron's position. A patron's order will not be activated for dispatch until he/she has been actually present at some waiting spot. Our server program has been coded in Python, same language used for robot code. Currently the Mots server can only instruct TurtleBot3 to move one-by-one, affecting the batched moving accuracy of our bump-free scheduling algorithm. The problem is likely caused by the robots' WiFi modules built in Raspberry Pi 3 being unable to successfully receive multiple instructions from Mots server under the carrier-sensing access model, resulting in serious packet collisions. By replacing the built-in WiFi modules with external high-performance TP-Link USB WiFi interface cards (TL-WN722 N) in our current prototype, the packet collisions problem has been noticeably improved, yet not fully solved when link-layer medium contention is high. We continue to figure out other solutions to resolving the communications inefficiency problem. Our goal is to ensure all smart robots scheduled in the same batch can depart at exactly the same time.

C. Real-Time Image Recognition

To enable complete automation of the takeout process, we incorporate AI-assisted image recognition functionality in our prototype. We study various learning techniques for recognizing numbers, which can be handwritten, printed, or digitally displayed. Several AI-based modules are readily available. For example, OCR (Optical Character Recognition), supported by OpenCV [14], is widely used for recognizing paper text. Another popular method used to recognize numbers and characters is the ANPR/ALPR (Automatic Number/License Plate Recognition) system [15], [16]. A commercial ALPR-based company can be found at [17]. For handwritten digits, the MNIST dataset provides a sufficient training set of 60000 examples and a test set of 10000 examples. In our prototype, we develop a two-layer CNN model based on MNIST dataset. To prevent the model from overfitting, 20% of the original training data was taken out as the validation dataset during the training process. We draw optimization and performance learning curves to observe the degree of training. Fig. 8(a), (b) shows that the learning model of 200-time training epochs exhibits an overfitting. By reducing the training epochs to 50-time, we obtain a better-fitting learning model. Therefore, we successfully recognize the Pi camera

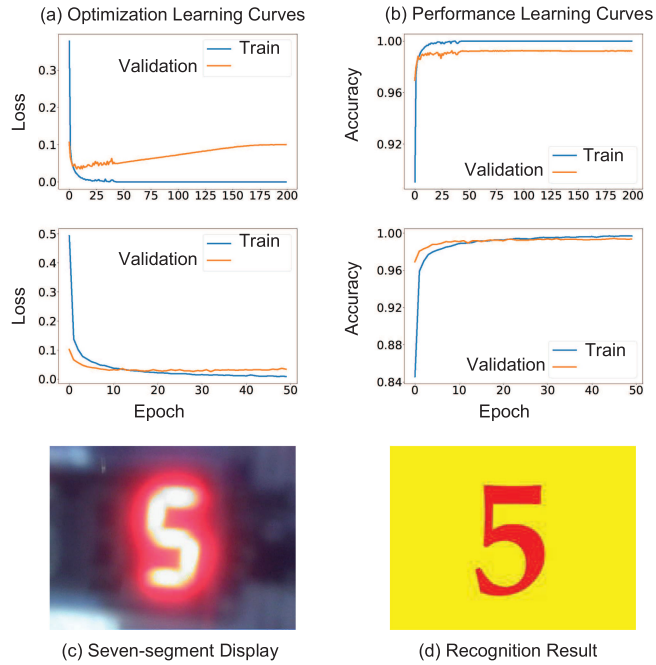


Fig. 8. AI-assited learning curves and CNN-based image (number) recognition result.

captured image (number) from the seven-segment display in real time. Fig. 8(c), (d) displays the recognition result. As we observe local catering businesses, most shop owners are used to hand-writing order numbers on paper. Through AI technology, both handwritten and screen digits can be instantly recognized based on proper dataset such as MNIST. We expect to promote the service automation not merely for international restaurant franchises but also among local, traditional mall shops to help them earn more business profits.

D. Mots in Action

By putting all technological pieces together, we have built a proof-of-concept Mots prototype realized by AI-assisted smart robots. This working prototype facilitates us to further investigate and evaluate service automation solutions in real life. A brief demonstration video of our Mots operations is available at [1].

VI. Conclusion and Future Applications

In this work, we propose an AI-assisted robotic solution that offers contactless meal order and takeout service (Mots) during the coronavirus pandemic. As keeping social distance has become a new normal, our system aims to minimize human contact while maintaining satisfactory dietary needs with service promptness and accuracy. For the post COVID-19 era, Mots also represents a useful IoT-based real-world solution. In the near future, we expect to leverage existing IoT development tools [18] to expand Mots capacity. Our bump-free algorithm, in particular, can be potentially applied in several related fields, such as WSN for scheduling collision-free travel paths of mobile sensors [19], [20] or massive (large-scale) automated warehouses for efficient order pick-up and dispatch applications in the automation industry [21], [22]. Despite many life disruptions brought by the pandemic, we believe this global crisis actually has been spurring and shepherding numerous theoretical research ideas into practical innovations.

References

[1] T.-Y. Lin, K.-R. Wu, Y.-S. Chen, W.-H. Huang, and Y.-T. Chen, "Demo: Contactless meal order and takeout service via AI-assisted smart robots," 2020. Accessed: Oct. 10, 2020. [Online]. Available: http://bun.cm.nctu.edu.tw/Mots/Mots-demo.mp4

[2] V. C. S. Yeo, S.-K. Goh, and S. Rezaei, "Consumer experiences, attitude and behavioral intention toward online food delivery (OFD) services," *J. Retailing Consum. Serv.*, vol. 35, pp. 150–162, Mar. 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int'l Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.

[5] T.-Y. Lin, H. A. Santoso, C.-A. Lin, and G.-L. Wang, "CFPP: Collision-free path planning for wireless mobile sensors deployment," in *Proc. IEEE Int'l Conf. Commun.*, Jun. 2015, pp. 6412–6417.

[6] D. J. A. Welsh and M. B. Powell, "Upper bound for the chromatic number of a graph and its application to timetabling problems," *Comput. J.*, vol. 10, p. 85–86, 1967.

[7] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, Apr. 1979.

[8] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *J. Res. Nat. Bur. Standards*, vol. 84, no. 6, pp. 489–505, Nov./Dec. 1979.

[9] "Specifications of TurtleBot3," 2020. Accessed: Oct. 10, 2020. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/

[10] "Teach, Learn, and M.with RaspberryPi," 2020. Accessed: Oct. 10, 2020. [Online]. Available: https://www.raspberrypi.org/

[11] "OpenCR: ROS Embedded Systems," 2020. Accessed: Oct. 10, 2020. [Online]. Available: https://emanual.robotis.com/docs/en/parts/controller/opencr10/

[12] "Raspberry Pi Camera Module," 2016. Accessed: Oct. 10, 2020. [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/

[13] T. M. Electronics, "LED drive control special circuit TM1637," 2015. Accessed: Oct. 10, 2020. [Online]. Available: https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_TM1637.pdf

[14] OpenCV, "Scene text recognition," 2020. Accessed: Oct. 10, 2020. [Online]. Available: https://docs.opencv.org/master/d8/df2/group__text__recognize.html

[15] S. Fakhar A. G., M. Saad H., A. Fauzan K., R. Affendi H., and M. Aidil A, "Development of portable automatic number plate recognition (ANPR) system on Raspberry Pi," *Int. J. Elect. Comput. Eng.*, vol. 9, no. 3, pp. 1805–1813, 2019.

[16] S. Khokhar and P. K. Dahiya, "Character Recognition for ALPR Systems: A new perspective," *Innovat. Electron. Commun. Eng.*, vol. 107, pp. 479–485, 2020.

[17] P. Recognizer, "License plate recognition - high accuracy ALPR," 2020. Accessed: Oct. 10, 2020. [Online]. Available: https://platerecognizer.com/

[18] Y.-B. Lin, Y.-W. Lin, C.-M. Huang, C.-Y. Chih, and P. Lin, "IoTtalk: A management platform for reconfigurable sensor devices," *IEEE Internet Things J.*, vol. 4, pp. 1552–1562, Oct. 2017.

[19] T.-Y. Lin, H. A. Santoso, and K.-R. Wu, "Global sensor deployment and local coverage-aware recovery schemes for smart environments," *IEEE Trans. Mobile Comput.*, vol. 14, no. 7, pp. 1382–1396, Jul. 2015.

[20] T.-Y. Lin, H. A. Santoso, K.-R. Wu, and G.-L. Wang, "Ehanced deployment algorithms for heterogeneous directional mobile sensors in a bounded monitoring area," *IEEE Trans. Mobile Comput.*, vol. 16, no. 3, pp. 744–758, Mar. 2017.

[21] T. van Gils, K. Ramaekers, A. C., and R. M. de Koster, "Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review," *Eur. J. Oper. Res.*, vol. 267, no. 1, pp. 1–15, May 2018.

[22] J. A. Cano, A. A. Correa-Espinal, and R. A. G. Montoya, "Evaluation of picking routing policies to improve warehouse efficiency," *Int. J. Ind. Eng. Manage.*, vol. 8, no. 4, pp. 229–238, Dec. 2017.