

# A scalable multi-robot task allocation algorithm

Chayan Sarkar, Himadri Sekhar Paul, Arindam Pal

TCS Research, Kolkata, India

{sarkar.chayan, himadriSekhar.paul, arindam.pall}@tcs.com

**Abstract**—In modern warehouses, robots are employed along with humans to perform those jobs that are less productive and unsafe for humans. This paper focuses on the tasks where robots fetch outgoing objects from their respective storage racks and bring them to the packaging dock. This requires a careful task allocation along with route planning such that the total path traveled (cost) by all the robots to complete all the tasks is minimized. The number of tasks that can be performed in a single run (part of the same route) depends on the maximum capacity of the robot and the combined weight of the objects on that route. This task allocation problem can be mapped to the capacity-constrained vehicle routing problem (CVRP), which is an NP-hard problem. Though there exist a number of algorithms that provide a near-optimal solution to a CVRP instance, they do not scale well with the number of nodes (tasks). Thus, we develop a fast heuristic algorithm, called nearest-neighbor based Clustering And Routing (nCAR) that provides a solution in a fraction of time compared to the state-of-the-art solution, and also reduces the cost of the solution when there are a large number of nodes. We compare the performance of nCAR with the Google OR-Tools and obtain a solution at 1/6-th of time for 2000 tasks. Though OR-Tools provides a low-cost solution for small number of nodes, it takes 1.5 times more execution time and routes to complete all the tasks, compared to nCAR.

## I. INTRODUCTION

The advent of mobile robot technology along with their affordability has increased the usability scope of the autonomous moving robot such as pet [28], guide [35], search and rescue operations [20], industrial automation [13], etc. The later two examples often constitutes a group of robots as opposed to a single robot in the earlier use-cases in order to speed up the task completion time. In general, automation using a group of mobile robots constitutes of two major steps – task allocation and task execution. The goal of task allocation is to ensure that the overall cost of executing all the tasks is minimized, while redundant robot resources are not allocated to complete a task. On the other hand, task execution ensures that the tasks are executed correctly and safely according to the assignment. In this article, we focus on task allocation.

Multi-robot task allocation (MRTA) is a well studied problem in the robotic algorithm domain. There are various class of problems within the MRTA domain with different complexity; thus there exists completely different solution approaches as well.

**Context.** The classical ST-SR (single-task robot and single-robot task) is a widely studied problem, and there exists a number of polynomial time solutions [15], [39] that reduces the problem to the classical assignment problem. However, there are tasks, mostly in industrial setup, that can not be

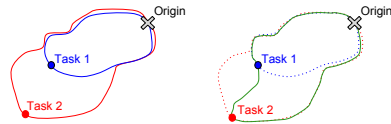


Fig. 1. Single task completion on a single run v/s more than one task completion on a single run.

completed by a single robot alone. In such cases, a team of robots is constituted and assigned to complete a single task. In such ST-MR (single-task robot and multi-robot task) assignment, the emphasis is on the constitution of a suitable team of robots to complete a particular task [16], [34]. On the other hand, MT-SR (multi-task robot and single-robot task) assignment problem, where one robot is assigned multiple tasks, are rarely studied due to their limited use-cases.

**Motivation.** In a warehouse scenario, the task of a robot is defined as moving to the location of a particular object, fetching the object from the storage rack, and returning to the original location (packaging dock). Though a robot almost never fetches multiple objects from the racks simultaneously, it can fetch more than one object on a single run, before returning back to the dock. This will reduce the overall travel time as compared to fetching one object on a single run.

Fig. 1 shows the ST-SR and MT-SR scenarios in a warehouse. Obviously, the total time required to fetch all the required objects is smaller in the later case if the objects in close proximity are fetched on a single run. The ST-SR is a special constrained version of the MT-SR problem. The associated problem in MT-SR case is to group tasks, optimally, to be given to a robot, given that each of the robots have a certain (fixed) carrying capacity.

**Problem description.** In this paper, we propose an MRTA mechanism for warehouse automation using a group of robots. We reduce the MT-SR assignment problem to a capacity-constrained vehicle<sup>1</sup> routing problem (CVRP), where each robot is constrained by its maximum weight carrying capacity. The problem is to assign tasks to the robots such that the overall time to bring the items to the packaging dock can be minimized.

CVRP is known to be an NP-hard problem [22]. Though there exists a number of exact [3], [4] and heuristic [23], [14] algorithms for CVRP, they do not scale well with the number of nodes (tasks) [5]. In the warehouse scenario, the number of nodes can be large and vary significantly (from couple of hundreds to thousands) as opposed to smaller number of

<sup>1</sup>In the rest of the paper, we use vehicle and robot interchangeably.

nodes in typical CVRP instances (generally upto a hundred). Thus, the prevalent exact algorithms becomes impractical. On the other hand, a smart heuristic algorithm is required which provides a low-order polynomial time complexity as well as a low-cost (close to optimal) solution.

**Our contributions.** In this paper, we present a fast heuristic for CVRP, called **n**ear**e**st-neighbor based **C**lustering **A**nd **R**outing (nCAR). We compared the performance of nCAR with the state-of-the-art algorithm provided by Google Optimization Tools (OR-Tools) and achieved better results. Specifically, our contributions are as follows.

- We developed an  $O(n^3)$  time heuristic algorithm, which gives a solution whose cost is at most 1.6 times the optimal solution for benchmark datasets (with smaller number of nodes), and achieves better result compared to the state-of-the-art algorithm (for test datasets with large number of nodes).
- The required number of routes to complete all the tasks (produced by nCAR) of a CVRP instance is close to the theoretical minimum, and outperforms the state-of-the-art solution with respect to the required number of routes and the time required to find a solution, irrespective of the number of nodes in the CVRP instance.

## II. RELATED WORKS

The goal of this work is to develop a task assignment mechanism in a warehouse where there are multiple robots, and each robots can pick multiple objects from various shelves located at different parts of the warehouse floor. The object assignment problem is studied under the Multi-Robot Task Assignment (or *MRTA*) abstraction. Once a set of objects is assigned to a robot, it needs to collect all of them, not necessarily following any pre-defined order, and carry them to a pre-defined location *i.e.*, a dock. This part of the problem is studied under the path planning problem. Therefore, this problem has at least two dependent components – *task assignment* and *planning the path of the robots*. The task assignment problem can be reduced to the *bin-packing problem* and the path planning problem can be reduced to the *travelling salesman problem*, both of which are known to be NP-hard.

Surynek *et al.* [33] showed that the path planning problem for multiple robots, under the constraint that the makespan of the travel duration of the robots have to be minimized is NP-hard. The authors of [29] present a subgraph decomposition based heuristic for path planning for multi-robot collaborative systems. The heuristic first partitions road maps into smaller ones, each of which can be modeled as a subgraph from a known set of structured subgraphs. This decomposition reduces a large road graph into a set of smaller road graphs, thereby effectively reducing the search space. The author modifies his heuristic for more practical road-maps structures in [30].

Authors in [24] present a distributed method of task assignment for multiple robots inspired by market-based coordination protocol. In a dynamic environment and continuous domain, the search space for path planning explodes.

*Rapidly-exploring Random Trees* (RRTs) [21] structure has been developed to efficiently manage and explore this explosion. Authors in [8] presents a randomized algorithm on RRTs, named *Execution extended RRT* (ERRT) for on-line path planning while path execution is in progress.

Exploratory robots, which move in partially known environment, may come across unexpected obstacles and need to bypass it. Such a feature is desirable, even in a known environment like that of a warehouse, where a robot may encounter another robot or a pile of boxes on its planned path. In such a partially known environment, the robot need to modify its paths for the observed obstacle only. The authors in [31] presents the  $D^*$  algorithm, which is capable of planning paths in unknown, partially known, and challenging environments in an efficient and optimal manner.

The conjunction of the two problems, *MRTA* and path planning, can be reduced to the *CVRP* problem. The general description of the *CVRP* includes a set of vehicles required to visit a set of locations. The locations have certain demands to be served. The vehicles act as carriers of the objects of these demands. The vehicles have limited carrying capacity and they need to serve demands of all of the sites. There are different optimization criterion around this general formulation. In the context of this warehouse automation related problem, we focus on the minimization of the number of robots required to serve all sites and also the total length of the path traversed by the robots.

The Vehicle Routing Problem (VRP) has a long history. It is a generalization of the well-known Traveling Salesman Problem (TSP). The problem was introduced by Dantzig and Ramser [12]. It is a *combinatorial optimization* and *integer programming* problem. The book by Toth and Vigo [36] gives an excellent treatment of the topic. This problem is known by several other names – *ORIENTEERING*, *DIAL-A-RIDE*, *DEADLINE-TSP* and *DISCOUNTED-REWARD-TSP* among others. We discuss some of the relevant works here.

Blum *et al.* [7] gave the first constant-factor approximation algorithm for the rooted *ORIENTEERING* problem, as well as a new problem that they called the *DISCOUNTED-REWARD-TSP*. Both of these problems were motivated by robot navigation. Bansal *et al.* [6] gave an  $O(\log n)$ -approximation algorithm for the *DEADLINE-TSP* problem, and an  $O(\log^2 n)$ -approximation algorithm for the *VEHICLE ROUTING WITH TIME-WINDOWS* problem. Chekuri *et al.* [9] gave a  $(2 + \epsilon)$ -approximation algorithm for *ORIENTEERING* in undirected graphs and an  $O(\log^2 OPT)$ -approximation algorithm for *ORIENTEERING* in directed graphs. Here  $OPT \leq n$  is the number of vertices visited by an optimal solution.

Christiaens *et al.* [10] designed a single ruin and recreate method to give a heuristic solution for CVRP. Subramanian *et al.* [32] proposed a hybrid algorithm for a class of vehicle Routing Problems with homogeneous fleet using a Mixed Integer Programming (MIP) solver. Vidal *et al.* [38] developed a efficient general framework for solving multi-attribute. Some of the other recent works include [18], [19], [27], [17], [26], [25].

In this paper, we present a heuristic algorithm running on a

centralized system, where the plan for each of the participant robots are generated and distributed for the robots to execute.

### III. PROBLEM FORMULATION

In a warehouse, an item is scheduled to be dispatched when a customer places an order for it. Usually an warehouse is spread across a very large area, and items are also stored in racks across this vast field. The job of a robot is to fetch the items from the storage racks to the packaging dock. In this paper, we assume that the robots are homogeneous and each one has a fixed maximum weight carrying capacity. As mentioned earlier, we assume the tasks are also homogeneous in nature, but they differ based on the storage location and weight of the item. In general, tasks are handled as a wave, where a wave can contains couple hundreds to thousands tasks. Once these tasks are completed, the next wave, which accumulates during the execution of the previous wave, is picked for execution.

In this paper, we propose a heuristic such that the cost of fetching all the ordered items can be minimized. Here the cost refers to the total path length traveled by all the robots to complete all the tasks. A robot always starts from and comes back to the packaging dock of the warehouse. On a single run, a robot can fetch multiple items, i.e., completing multiple tasks, provided the total demand (weight) of all the tasks is within the capacity of the robot. Thus, it can be classified as a MT-SR (multi-task robot and single-robot task) assignment problem.

The problem can be reduced to the classical capacity-constrained vehicle routing problem (CVRP), where a number of vehicles start and end their journey at a depot. On their journey, they service a number of customers, where the total demand of all the customers on a single journey must be within the capacity limit of the robots. The goal is to find the optimal route for each of the robots such that the total cost of the all the journeys made by this fleet of robots is minimized and all the customers are serviced by exactly one robot. As CVRP is known to be NP-hard [36], the MT-SR assignment problem is also an NP-hard problem.

#### A. Mathematical formulation

Let  $G = (V, E)$  be an undirected, complete graph. The number of vertices in this graph is  $|V| = n+1$ . We denote the vertices as  $V = \{v_0, v_1, \dots, v_n\}$ . Here, the special vertex  $v_0$  corresponds to the *depot* and the other vertices  $V' = V \setminus \{v_0\}$  correspond to the  $n$  *sites*. Let  $V_i = \{v_i\}, 0 \leq i \leq n$ .

Each edge  $e \in E$  has a *cost*  $c_e > 0$ . This is the cost of traveling the edge  $e$ . For each vertex  $v_i \in V'$ , there is an associated *demand*  $d_i > 0$ , which has to be satisfied. Let  $k$  be the *number of robots* available at the depot and let  $C$  be the *capacity* of each robot. We assume without loss of generality that the maximum demand of any vertex is at most the capacity of each robot, i.e.,  $d_i \leq C$ , for all  $v_i \in V'$ .

Our goal is to find a set of vertex-disjoint cycles (except at the depot) such that the total cost of the cycles is minimized and the total demand of all the vertices in each cycle does

not exceed the capacity of any robot. Mathematically, we want to find  $t$  cycles  $C_1, \dots, C_t$  such that:

- 1)  $\sum_{i=1}^t c(C_i)$  is minimized.
- 2)  $C_i \cap C_j = \{v_0\}$ , for  $1 \leq i < j \leq t$ .
- 3)  $\sum_{v_j \in C_i} d_j \leq C$ , for  $1 \leq i \leq t$ .

The cost of a cycle  $C$  is  $c(C) = \sum_{v \in C} c(v)$ .

### IV. ALGORITHM DESIGN

Since CVRP is known to be an NP-hard problem [36], it is unlikely to have a polynomial-time algorithm, unless  $P = NP$ . Thus, various heuristic algorithms are developed to solve it in polynomial time. However, the existing algorithms can produce a near-optimal solution only for a small number of nodes. In a modern warehouse application scenario, the number of nodes are significantly higher, ranging from couple of hundreds to thousands. This motivates us to develop a heuristic algorithm, which can produce a fast solution to a CVRP instance having a large number of nodes. In Algorithm 1, we present our heuristic algorithm *nearest-neighbor based Clustering and Routing* (nCAR) to solve CVRP. Note that nCAR is not applicable for the warehouse scenario only. It can also be utilized for any application domain, where the CVRP instances contain a large number of nodes.

Given the maximum capacity of the vehicles ( $C$ ) and a set of nodes ( $N$ ), with their location and demand specified, nCAR provides the number of vehicles required to visit all the nodes ( $v_{count}$ ). Additionally, it provides the route of each vehicle as a sequence of nodes ( $R_{list}$ ) and the total cost to travel all the routes ( $c_{total}$ ). The objective of nCAR is to minimize the total path cost while finding the list of routes, where every route fulfills the capacity constraints of all the vehicles, and so can be assigned to the vehicles.

The heuristic nCAR splits all the nodes into a number of clusters, where clusters contain disjoint set of nodes except the depot. Then a route is constructed for each of the clusters by mapping it to a traveling salesman problem (TSP). Given a set of nodes  $T$ ,  $TSP(T)$  returns a Hamiltonian cycle (route)  $R$  and it's cost  $c$ . nCAR uses the best known approximation algorithm for TSP by Christofides [11], whose approximation ratio is 1.5. The criteria to create a cluster is two-fold:

- The nodes within a cluster should form a *feasible route*, i.e., the total demand of all the cluster members are within the capacity constraint ( $C$ ) of the vehicles.
- The clusters should be formed in such a way that the path cost of individual clusters lead to a lower total path cost.

The algorithm starts with all the nodes as unassigned, i.e., not part of any cluster/route. It iteratively finds a cluster (feasible route) at every round. The algorithm continues until all the nodes are assigned to a cluster. In a round, the yet to be assigned (to a cluster) nodes are split into two parts – a cluster of nodes that can form a feasible route ( $T$ ) and the remaining nodes ( $\bar{T}$ ) by using the function `feasibleRoute`. The algorithm continues until  $\bar{T}$  is empty. As mentioned earlier, `feasibleRoute` finds a new cluster of nodes at

---

**Algorithm 1:** Nearest-neighbor based Clustering and Routing (*nCAR*): A fast algorithm for capacity-constrained vehicle routing problem (CVRP).

---

**Input:** A graph  $G = (V, E)$ , demands of each vertex ( $d_i$ ), and the capacity of the vehicles ( $C$ ).

**Output:** The required number of vehicles ( $v_{count}$ ), the list of routes ( $R_{list}$ ), and the total cost of all the routes ( $c_{total}$ ).

```

1 Algorithm nCAR ( $V, C$ )
2    $N \leftarrow V$ ;
3    $v_{count} \leftarrow 0$ ;
4    $c_{total} \leftarrow 0$ ;
5    $R_{list} \leftarrow \emptyset$ ;
6   while ( $\neg N.empty()$ ) do
7      $[T, \bar{T}] \leftarrow feasibleRoute(N, C)$ ;
8      $[R, c] \leftarrow TSP(T)$ ;
9      $R_{list} \leftarrow R_{list} \cup R$ ;
10     $v_{count} \leftarrow v_{count} + 1$ ;
11     $c_{total} \leftarrow c_{total} + c$ ;
12     $N \leftarrow \bar{T}$ ;
13  return  $[R_{list}, v_{count}, c_{total}]$ ;

1 Procedure feasibleRoute ( $N, C$ )
2    $p_{min} \leftarrow singleNodeCycle(N)$ ;
3   for ( $i = 1 : |N|$ ) do
4      $N_T \leftarrow \{N[0], N[i]\}$ ;
5      $\bar{N}_T \leftarrow N \setminus \{N[i]\}$ ;
6      $d_T \leftarrow N[i].d$ ;
7      $k \leftarrow i$ ;
8     while ( $d_T < C$ ) do
9        $k \leftarrow$ 
10       $restrictedNearestNeighbor(k, \bar{N}_T, d_T)$ ;
11      if ( $k = 0$ ) then
12        break;
13       $d_T \leftarrow d_T + \bar{N}_T[k].d$ ;
14       $N_T \leftarrow N_T \cup \bar{N}_T[k]$ ;
15       $\bar{N}_T \leftarrow \bar{N}_T \setminus \bar{N}_T[k]$ ;
16     $p \leftarrow eulerCycle(N_T) +$ 
17     $singleNodeCycle(\bar{N}_T)$ ;
18    if ( $p < p_{min}$ ) then
19       $p_{min} \leftarrow p$ ;
20       $T_{min} \leftarrow N_T$ ;
21       $\bar{T}_{min} \leftarrow \bar{N}_T$ ;
22  return  $[T_{min}, \bar{T}_{min}]$ ;

```

---

every round from the set of unassigned nodes. If there are  $N$  unassigned nodes, it creates  $N$  potential clusters that satisfy the feasibility criteria. Then, based on a greedy approach, it selects the best cluster out of these  $N$  potential clusters.

The  $i^{th}$  potential clusters is formed as follows. First, the  $i^{th}$  node becomes part of the ( $i^{th}$ ) cluster, and the cluster demand is initialized to the demand of the  $i^{th}$  node. Then, the  $i^{th}$  node becomes the current node. Next, the nearest neighbor of the current node is found based on the Euclidean distance. However, this nearest neighbor is selected with the restriction that the cluster's demand plus the neighbor's demand is within the capacity limit (*restrictedNearestNeighbor*). This ensures the feasibility of the cluster. If no such neighbor is found, the  $i^{th}$  cluster formation ends. On the other hand, if a neighbor is found with the capacity restriction, it becomes part of the cluster and also the current node, and its demand is added to the cluster demand. In this way, the procedure continues to find the restricted nearest neighbor of the current node.

To select the best cluster out of these  $N$  potential clusters such that the total path cost of all the routes can be minimized, a greedy approach is used. For every potential cluster, there is also a set of unassigned nodes (which can be empty as well). An Euler cycle and a single node cycle (loop) costs are calculated for the cluster nodes and unassigned nodes, respectively. The summation of these two cycle cost is associated to the potential cluster. The best cluster is selected that has the lowest associated cost.

#### A. Time-complexity analysis

Let us first consider the procedure *feasibleRoute*, which constructs one feasible route. Both the sets  $N_T$  and  $\bar{N}_T$  are constructed from the vertex set  $V$  (lines 4, 5, 13 and 14). Hence,  $|N_T| < n$  and  $|\bar{N}_T| < n$ , where  $n = |V|$ . The value of  $d_T$  monotonically increases within the *while* loop at line 8. The incremental value is computed at line 12, and the value is the cost of one of the nodes from the set  $\bar{N}_T$ . The chosen  $k^{th}$  node is removed from  $\bar{N}_T$ . This *while* loop is terminated when, either  $d_T$  exceeds the capacity  $C$ , or all feasible nodes in  $\bar{N}_T$  are exhausted. In either case, the *while* loop is executed at most  $\min\{C, |\bar{N}_T|\} < n$  times. Therefore, the cost of this *while* loop is  $O(n)$ .

The procedure *eulerCycle* is based on the linear-time depth-first search (DFS) of the nodes in  $V$ . So, its running time is  $O(n)$ . Both the call to procedure *eulerCycle* and the *while* loop at line 8, are contained in a *for* loop at line 3. This outer *for* loop is bounded by  $|N|$ , i.e., the complexity of this loop is  $O(n)$ . Therefore, the complexity of the function *feasibleRoute* as a whole is  $O(n \times (O(n) + O(n))) = O(n^2)$ .

Now let us consider the main function *nCAR*, where the function *feasibleRoute* returns one non-empty route on each invocation, i.e., until  $T \neq \emptyset$ . The set  $N$  is initialized as  $V$  (line 2) and  $T$  is removed from  $N$  on line 12 at each iteration of the *while* loop (line 6). Therefore, the set  $N$  monotonically shrinks in size. This implies that the *while* loop at line 6 is executed at most  $|V|$  times, which is  $O(n)$ .

TABLE I

PERFORMANCE COMPARISON AMONG THE OPTIMAL, GOOGLE OR-TOOLS, AND nCAR BASED ON A P-SET OF CVRPLIB [1]. SOME OF THE EXCEPTIONAL CASES ARE HIGHLIGHTED, E.G. OR-TOOLS PRODUCES LEAST COST SOLUTION (RED), OPTIMAL SOLUTION HAVE MORE ROUTES (BLUE).

dataset	Optimal		OR-Tools		nCAR	
	cost	routes	cost	routes	cost	routes
P-n16-k8	450	8	450	8	450	8
P-n19-k2	212	2	230	2	247	2
P-n20-k2	216	2	227	2	247	2
P-n21-k2	211	2	211	2	254	2
P-n22-k2	216	2	220	2	278	2
P-n22-k8	<b>603</b>	<b>8</b>	<b>595</b>	<b>9</b>	<b>797</b>	<b>8</b>
P-n23-k8	529	8	538	9	605	10
P-n40-k5	458	5	459	5	670	5
P-n45-k5	510	5	543	5	719	5
P-n50-k7	554	7	575	7	781	7
P-n50-k8	631	8	681	9	750	9
P-n50-k10	696	10	778	10	868	10
P-n51-k10	741	10	838	10	1014	11
P-n55-k7	568	7	579	7	772	7
P-n55-k8	<b>588</b>	<b>8</b>	<b>627</b>	<b>7</b>	<b>759</b>	<b>7</b>
P-n55-k10	694	10	735	10	1044	10
P-n55-k15	989	15	977	16	1156	16
P-n60-k10	744	10	844	10	877	10
P-n60-k15	968	15	1025	15	1149	15
P-n65-k10	792	10	875	10	1080	10
P-n70-k10	827	10	923	10	1021	10
P-n76-k4	593	4	678	4	745	4
P-n76-k5	627	5	712	5	798	5
P-n101-k4	681	4	793	4	980	4

Since feasibleRoute is of  $O(n^2)$ , running time of nCAR is  $O(n \times n^2) = O(n^3)$ .

## V. EVALUATION

In this section, we evaluate the performance of nCAR from three aspects – (i) cost of the solution, (ii) number of routes in the solution, and (iii) execution time of the algorithm. The cost of the solution is measured as the total distance traveled by all the vehicles to visit all the nodes. Since a single vehicle would not be able to visit all the nodes in one go (due to capacity constraint), the solution comprises of a number of routes, where a route comprises of a sequence of nodes. If there are as many vehicles available as the number of routes in the solution, all the routes can be traversed in parallel by different vehicles. However, if the number of vehicles is less than the number of routes, the assignment becomes time-extended and a vehicle starts a new route after completing the previously assigned route. Whether each route is traversed by a different vehicle or a vehicle traverses multiple routes, the cost of the solution remains unchanged. Also, the sequence of traversing the routes do not have any effect on the solution cost. The third performance metric is the absolute running time (in milliseconds) of the algorithm on a standard computer (Intel 5th generation processor with dual core and 4 GB memory).

### A. Comparison of nCAR with the Optimal solution

First, we evaluate the performance of nCAR for some well-known instances of CVRP (benchmarks) whose optimal

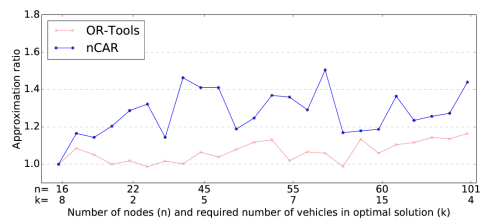


Fig. 2. Cost ratio of OR-Tools and nCAR with the optimal solution for the P-dataset of CVRP benchmarks.

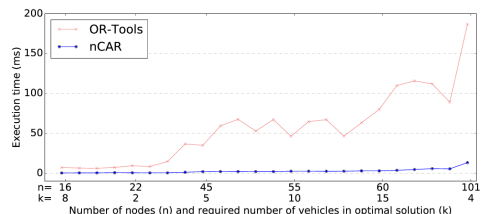


Fig. 3. The execution time to find the solution using OR-Tools and nCAR for the P-dataset.

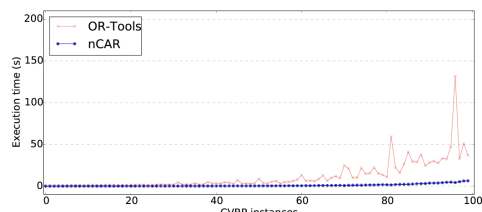


Fig. 4. The execution time comparison between OR-Tools and nCAR for the X-dataset.

solutions are known. The benchmark datasets are available in [1]. Here we present the results for the P-dataset whose optimal solution are provides by Augerat *et al.* [3]. Table I summarizes the cost and the number of routes for these CVRP instances that are solved optimally, the state-of-the-art Google Optimization Toolbox (OR-Tools) [2], and our algorithm (nCAR). Solutions provided by both OR-Tools and nCAR are close to the optimal solutions, as evident from Table I. Moreover, Figure 2 shows the cost ratio of OR-Tools and nCAR, where cost ratio is defined as,

$$\text{cost ratio} = \frac{\text{cost of the heuristic algorithm}}{\text{cost of the optimal algorithm}}.$$

We notice two peculiar instances (P-n22-k8 and P-n55-k15), where the cost ratio of OR-Tools is less than 1. In other words, OR-Tools finds a lower cost solution than the optimal solution, which is impossible in theory. We have verified the total cost of the solution of these instance as provided by the optimal solution and the OR-Tools. These discrepancies are mainly due to floating point rounding-off of the Euclidean distances. However, for the instance P-n22-k8, the OR-Tools always provide a low cost solution. Though the optimal solution requires fewer routes, the cost of the solution is higher, which is the only optimization goal without any restriction on the required number of routes. This invalidates the optimal solution for this particular instance.

Though the required number of vehicles for all the three

algorithms are almost same for most of the instances, the optimal solution often comprises of less number of routes. However, there is one instance (P-n55-k8) where the optimal solution comprises more routes as compared to the other two algorithms. As the goal of the algorithm is to minimize the total cost, this situation is definitely possible since the cost of the optimal solution is lower than the other two.

Even though OR-Tools often has a much lower cost ratio as compared to nCAR, it requires much higher computation time as evident from Figure 3. This holds true across the dataset. For example, a similar result is shown in Figure 4 for newest dataset provided by Uchoa *et al.* [37] that consists of 100 to 1000 nodes.

### B. Performance of nCAR for large number of nodes

As the given MT-SR assignment problem (warehouse scenario) needs to be solved when there is relatively large number of nodes (of the order of hundreds and thousands), we also analyze the performance of nCAR for such large instances. As the optimal solutions for such instances can not be produced in a reasonable time, we compare nCAR with the solutions produced by OR-tools only. Though the OR-Tools produces a low cost solution as compared to nCAR for the majority of the CVRP instances of small number of nodes, nCAR surpasses OR-Tools with respect to all the three parameters when CVRP instances of large number of nodes are used. Thus, in a warehouse scenario where the CVRP instances contain a large number of nodes, nCAR is a much better option.

Figure 5, 6, and 7 compares nCAR with OR-Tools with respect to the cost of the solution, execution time to find the solution, and the number of routes in the solution, respectively. The CVRP instances contain three different number of nodes, i.e., 500, 1000, and 2000. For each of these node sets, the minimum number of routes (theoretical) is varied based on the demand of each vehicle and capacity of each vehicle. Thus, there are a total of nine CVRP instances.

From Figure 5, it is evident that nCAR provides a low cost solution as the number of nodes increases. Moreover, OR-Tools requires a significantly higher execution time to find a solution. From Figure 6, it is clear that the execution time increases many fold with the number of nodes in the CVRP instance. The effectiveness of nCAR magnifies, if the cost of the solution is measured in terms of makespan to complete all the tasks. If there are  $r$  routes in the solution and there are  $r$  vehicles, all vehicles can be assigned one route each. The cost of the solution can be measured in terms of the maximum route length. Obviously, when there are more number of nodes, the solution also contains a significantly larger number of routes. Due to the limited number of vehicles, all the routes can not be visited in parallel. As a result, the assignment becomes time extended and a vehicle takes up a new route after completing the previous route. In this case, the cost of the solution is the maximum amount of time among all the vehicles. Thus, the lesser the number of routes, the better it is.

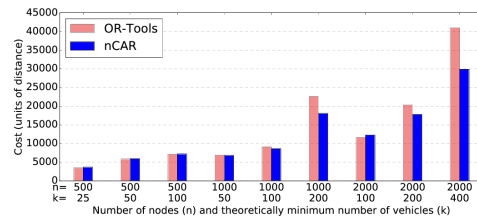


Fig. 5. Cost (total distance) comparison between nCAR and OR-Tools for large number of nodes.

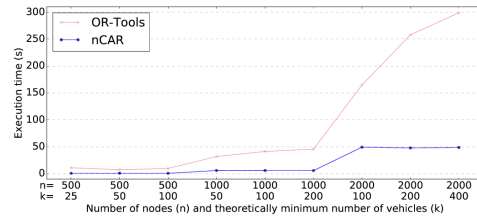


Fig. 6. Performance comparison between nCAR and OR-Tools for large number of nodes in terms of execution time to find the solution.

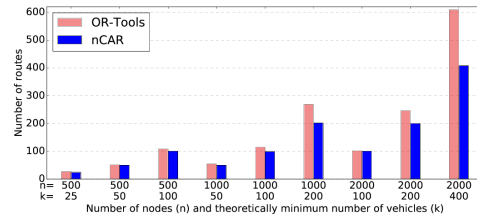


Fig. 7. Performance comparison between nCAR and OR-Tools for large number of nodes in terms of number of routes.

## VI. CONCLUSION

In this paper, we designed a multi-robot task allocation (MRTA) algorithm specifically keeping in mind a warehouse scenario, where a task is defined as picking up an object from its respective storage rack. Though a robot can pick multiple objects on a single run, the number of objects is constrained by its capacity. This problem can be reduced to the capacity-constrained vehicle routing problem (CVRP). Since CVRP is known to be NP-hard, this problem is also NP-hard. The state-of-the-art integer programming based optimal algorithms can solve CVRP instances upto 100 nodes. However, in a warehouse scenario, the number of nodes can be in the range of several hundreds to several thousands within an order wave. Thus, we developed a heuristic algorithm, called nearest-neighbor based Clustering And Routing (nCAR) that can provide a reasonably good result even if there are a large number of nodes. Experimental results show that our algorithm provides solutions that are close to the optimal solutions for smaller instances (cost ratio up to 1.6). For CVRPs with large number of nodes, where optimal solution is not available, we compare nCAR with the state-of-the-art approximation algorithm provided by the Google Optimization Toolbox (OR-Tools). Our evaluation shows that nCAR achieves a superior result as compared to

the OR-Tools in terms of the cost of the solution, the number routes, and the actual execution time of the algorithm.

## REFERENCES

- [1] Capacitated Vehicle Routing Problem Library. <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/>. Accessed on: 2017-09-01.
- [2] Google Optimization Tools (OR-Tools). <https://developers.google.com/optimization/>. Accessed on: 2017-03-30.
- [3] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. 1998.
- [4] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.
- [5] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.
- [6] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 166–174, 2004.
- [7] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007.
- [8] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388. IEEE, 2002.
- [9] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):23, 2012.
- [10] J. Christiaens and G. Vanden Berghe. A fresh ruin & recreate implementation for the capacitated vehicle routing problem. 2016.
- [11] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.
- [12] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.
- [13] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi. Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy. In *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, pages 209–215. IEEE, 2013.
- [14] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.
- [15] B. P. Gerkey and M. J. Mataric. Sold!: Auction methods for multi-robot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.
- [16] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.
- [17] I. L. Gørtz, M. Molinaro, V. Nagarajan, and R. Ravi. Capacitated vehicle routing with non-uniform speeds. In *Integer Programming and Combinatorial Optimization - 15th International Conference, IPCO 2011, New York, NY, USA, June 15-17, 2011. Proceedings*, pages 235–247, 2011.
- [18] I. L. Gørtz, M. Molinaro, V. Nagarajan, and R. Ravi. Capacitated vehicle routing with nonuniform speeds. *Math. Oper. Res.*, 41(1):318–331, 2016.
- [19] I. L. Gørtz, V. Nagarajan, and R. Ravi. Minimum makespan multi-vehicle dial-a-ride. *ACM Trans. Algorithms*, 11(3):23:1–23:29, 2015.
- [20] J. S. Jennings, G. Whelan, and W. F. Evans. Cooperative search and rescue with a team of mobile robots. In *Advanced Robotics, 1997. ICAR'97. Proceedings., 8th International Conference on*, pages 193–200. IEEE, 1997.
- [21] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.
- [22] J. K. Lenstra and A. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.
- [23] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.
- [24] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 128–133. IEEE, 2008.
- [25] V. Nagarajan and R. Ravi. Minimum vehicle routing with a common deadline. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 212–223, 2006.
- [26] V. Nagarajan and R. Ravi. Poly-logarithmic approximation algorithms for directed vehicle routing problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, pages 257–270, 2007.
- [27] V. Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.
- [28] J. Rhee. Intelligent pet robot, Mar. 12 2001. US Patent App. 09/802,960.
- [29] M. Ryan. Multi-robot path planning with subgraphs. In *Proceedings of the 19th Australasian Conference on Robotics and Automation, Auckland, New Zealand, Australian Robotics & Automation Association*. Citeseer, 2006.
- [30] M. R. Ryan. Graph decomposition for efficient multi-robot path planning. In *IJCAI*, pages 2003–2008, 2007.
- [31] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE, 1994.
- [32] A. Subramanian, E. Uchoa, and L. S. Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531, 2013.
- [33] P. Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI'10*, pages 1261–1263. AAAI Press, 2010.
- [34] F. Tang and L. E. Parker. A complete methodology for generating multi-robot task solutions using asymptre-d and market-based task allocation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3351–3358. IEEE, 2007.
- [35] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al. Minerva: A second-generation museum tour-guide robot. In *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, volume 3. IEEE, 1999.
- [36] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.
- [37] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.
- [38] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.
- [39] R. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.