Exact and Heuristic Approaches for the Multi-Agent Orienteering Problem with Capacity Constraints

Wenjie Wang, Hoong Chuin Lau * , and Shih-Fen Cheng
School of Information Systems, Singapore Management University, Singapore
{wjwang, hclau, sfcheng}@smu.edu.sg

***Abstract*—This paper introduces and addresses a new multi-agent variant of the orienteering problem (OP), namely the multi-agent orienteering problem with capacity constraints (MAOPCC). Different from the existing variants of OP, MAOPCC allows a group of visitors to concurrently visit a node but limits the number of visitors simultaneously being served at each node. In this work, we solve MAOPCC in a centralized manner and optimize the total collected rewards of all agents. A branch and bound algorithm is first proposed to find an optimal MAOPCC solution. Since finding an optimal solution for MAOPCC can become intractable as the number of vertices and agents increases, a computationally efficient sequential algorithm that sacrifices the solution quality is then proposed. Finally, a probabilistic iterated local search algorithm is developed to find a sufficiently good solution in a reasonable time. Our experimental results show that the latter strikes a good tradeoff between the solution quality and the computational time incurred.**

I. Introduction

Orienteering problem (OP) is a specialized single-agent routing problem whose aim is to produce a sequence of nodes to visit so that the collected reward from visited nodes is maximized, while the total travel time is bounded by a given time budget. Many interesting OP variants have been proposed over the years for different applications (a recent survey can be found in [1]). One of the contributions in this paper is to extend the OP literature by introducing a multi-agent orienteering problem with capacity constraints (MAOPCC). This problem is inspired by real-world applications in areas such as generating itineraries for visitors in leisure settings, e.g., in theme parks, cities, or museums (as highlighted in [2]). For ease of explanation, we will use itinerary generation in a theme park setting as the major example throughout the paper.

Figure 1 provides an example of a theme park where each attraction can only simultaneously serve a limited number of visitors each time (which contributes to capacity constraints). Each visitor could potentially have heterogeneous preference over attractions, and each visitor could also have visitor-specific time budget. And starting from the entrance, they will try to visit as many preferred attractions as possible in a limited time like one day. The goal is to develop a centralized planning algorithm that has the capability to generate itineraries quickly and collectively for a large number of visitors, where these itineraries are mutually dependent in terms of their waiting times in attractions.

*Corresponding Author

Compared to past OP literature, MAOPCC has the following unique features: 1) there are multiple agents in MAOPCC, whose goal is to collectively maximize total collected rewards (in our model, a node can be visited by multiple agents concurrently, and a distinct reward can be collected for each agent's visit; this is different from the team orienteering problem [3], in which a single agent takes multiple tours in separate time and a node can be visited at most once by one of the tours), 2) visiting a node incurs service time, and there are node-specific capacity constraints defined for all nodes (queuing is modelled exactly, which makes our model different from the one proposed in [2], where a worst-case assumption was made in estimating queuing time).

After describing the MAOPCC model, we will first introduce an exact approach using a branch and bound algorithm. However, as the exact approach is exponential in the number of agents and nodes (and thus intractable for even instances with moderate sizes). Since it is desirable in our problem to generate real-time guidance for thousands of users (though not all simultaneously), we propose a computationally fast sequential algorithm that finds a sub-optimal solution. To deal with the issue of sub-optimality, we then propose a local search algorithm that can find sufficiently good solutions in a reasonable time.

II. Related Work

Many approaches have been proposed to solve a wide variety of orienteering problems. [4] and [5] proposed exact approaches for solving OPs optimally. However, since OP is NP-hard [6], exact approaches ([4], [5]) can only solve OP instances with size up to hundreds of vertices. [7] and [8] proposed heuristics to approximate optimal OP solutions. [8] proposed the use of a greedy randomized adaptive search procedure (also known as GRASP), which generates initial solutions using four construction methods, followed by a two-phase local search procedure using exchange and insertion operations. Path relinking is also employed to further improve the GRASP performance.

The orienteering problem with time windows (OPTW) is an important class of OP which imposes a feasible time interval during which a particular node can be visited. OPTW can be solved heuristically [9] and exactly [10]. [9] first described an insertion heuristic that iteratively inserts a selected node with the highest ratio of score to shift time into the best available slot of the route and then proposed a tree heuristic that did a

Fig. 1. Illustrative example in a theme park.

depth-first search to construct a partial route beginning with the starting vertex. Partial routes are abandoned if they are infeasible or are not likely to produce a better route. On the other hand, [10] proposed an exact optimization method based on a dynamic programming model. Their approach performs a forward and backward search in a state search space by adding exact vertices beginning from the starting and end vertices respectively. A decremental state space relaxation strategy [11] is then employed so as to reduce the number of states.

The team orienteering problem (TOP) is the major OP variant that allows multiple agents to jointly traverse a network of nodes for rewards. A major different between our MAOPCC variant and TOP is that in TOP, each node can be visited at most once for rewards (in other words, if multiple agents visit a particular node, reward of this node is only counted once). The first heuristic for solving TOP was proposed by [12], which is derived from their previous work on single-agent OP [7]. [13] proposed a branch and price method for TOP that utilizes column generation during the branching phase. Heuristic tree-search is also implemented to improve performance.

Finally, the above two variants are combined to form a new variant called team orienteering problem with time windows (TOPTW). An iterated local search (ILS) was proposed to heuristically solve TOPTW [14]. The heuristic we develop in this paper is inspired by the one proposed in [14], yet with modifications necessary for the new features we introduce.

As far as multi-agent version of the OP is concerned, the first work was found in [2]. There, the authors considered the setting where agents are self-interested, namely, it is a variant of the congestion game; rather than an optimal solution, the concern is to find an equilibrium solution. In that work, an approach based on sampled fictitious play was proposed. In that work, the agent queuing sequence was not considered.

III. Problem Definition

We are given m agents and n vertices (attraction nodes) in a weighted graph G, where each agent A_k ($k \in \{1, ..., m\}$) has heterogeneous preference over the attraction nodes. Each node v_i ($i \in \{2, ..., n-1\}$) has the same service duration e_i and allows concurrent visits by multiple agents. It has a capacity c_i such that at most c_i agents can be served at the node v_i concurrently. When more than c_i agents visit node v_i concurrently at any time t, the policy of first-come first-served is employed, and an agent A_k who cannot be served immediately will have to wait in a queue and lead to a waiting time w_{ik}^t at node v_i. Assume an agent-specific reward s_{ik} is used to indicate the preference of the agent A_k over the attraction node v_i. Thus, the MAOPCC problem is described as finding time feasible routes for all agents with the maximum total rewards such that each agent A_k starts its movement from an initial node v_1 at time step T_k^1, visits a subset of the attraction nodes v_i ($i \in \{2, ..., n-1\}$) once and finally reaches a destination node v_n before the time step T_k^n.

Assuming that the travel time between two nodes v_i and v_j is t_{ij}. Let q_{ik}^t denote the queue sequence number of agent A_k at node v_i, i.e. it represents the total number of agents who have not departed from v_i and are or will be served before A_k. Let x_{ijk}^t be a binary decision variable that is set to 1 if agent A_k leaves v_i at time t and proceeds to visit v_j. Note that $x_{iik}^t = 0$ for $\forall(i, k, t)$ where $i \in \{1, ..., n\}$, $k \in \{1, ..., m\}$ and $t \in \{1, ..., T\}$. Let y_{ik}^t be the queuing sequence of agent A_k at node v_i when A_k arrives at v_i at time t, which means the number of agents who arrive at the node v_i at time t and queue before A_k. If A_k has not arrived at v_i yet or is being served at time t, then y_{ik}^t will be set to 0.

The mathematical model of MAOPCC is given by:

$$\max\ F(x) = \sum_{t=1}^{T}\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ik} x_{ijk}^t \tag{1}$$

subject to $\forall k \in \{1, ..., m\}, d \in \{2, ..., n-1\}$:

$$\sum_{t=1}^{T}\sum_{j=1}^{n} x_{1jk}^t = \sum_{t=1}^{T}\sum_{i=1}^{n} x_{ink}^t = 1 \tag{2}$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n} x_{idk}^{t} = \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \text{ and } \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \leq 1 \quad (3)$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n}(t + t_{id} + w_{dk}^{t+t_{id}} + e_d)x_{idk}^{t} = \sum_{t=1}^{T} t(\sum_{j=1}^{n} x_{djk}^{t}) \quad (4)$$

$$\sum_{t=1}^{T}\sum_{j=1}^{n} t * x_{1jk}^{t} = T_k^1 \text{ and } \sum_{t=1}^{T}\sum_{i=1}^{n} t * x_{ink}^{t} \leq T_k^n \quad (5)$$

$\forall k \in \{1, ..., m\}, d \in \{2, ..., n-1\}, t \in \{1, ..., T\}$:

$$q_{dk}^{t} = q_{dk}^{t-1} - \sum_{a=1}^{k-1}\sum_{j=1}^{n} x_{dja}^{t} - \sum_{a=k+1}^{m}\sum_{j=1}^{n} x_{dja}^{t} + y_{dk}^{t} \quad (6)$$

$$(y_{da}^{t} - y_{db}^{t})^2 \geq \sum_{i=1}^{n} x_{ida}^{t-t_{id}} \sum_{i=1}^{n} x_{idb}^{t-t_{id}} \forall a, b \in \{1, ..., k\}, a \neq b, \quad (7)$$

$$0 \leq y_{dk}^{t} < \sum_{i=1}^{n} x_{idk}^{t-t_{id}} \sum_{k=1}^{m}\sum_{i=1}^{n} x_{idk}^{t-t_{id}} \text{ and } y_{ik}^{t} \in Z \quad (8)$$

$$x_{iik}^{t} = 0 \text{ and } x_{ijk}^{t} \in \{0, 1\} \forall i, j \in \{1, ..., n\} \quad (9)$$

where $T = max_k(T_k^n)$. The waiting time w_{ik}^{t} for agent A_k at node v_i in function (4) can be computed as

$$w_{ik}^{t} = \sum_{\tau=t}^{t+\varepsilon^t e_i} z_{ik}^{\tau} \quad (10)$$

where $\varepsilon^t = \lfloor (q_{ik}^{t})/c_i \rfloor$, z_{ik}^{t} is a binary state variable that is set to 1 if agent A_k waits to be served at v_i at time $t (z_{ik}^{\tau} = 1$ if $\varepsilon^{\tau} > 0)$. The objective function (1) is the sum of total rewards collected by all agents. Constraint (2) ensures that each agent A_k starts its trip at v_1 and ends at v_n. Constraint (3) preserves the flow consistency at v_d and guarantees that all nodes are visited at most once. Constraint (4) ensures that arrival and departure times at all nodes are consistent at all nodes. Constraint (5) ensures that agent A_k leaves the starting node v_1 at time T_k^1 and reaches the destination node v_n before T_k^n. Constraint (6) models the queuing sequence before agent A_k at v_d in time t. Constraint (7) ensures that serving sequences are consistent among all agents at v_d at time t.

Solving this model directly is very challenging computationally, as it is nonlinear and contains integer variables. In the following, we will introduce three solution approaches, an exact method based on branch and bound, and two heuristic methods.

IV. Approaches for Multi-Agent Orienteering Problem With Capacity Constraints

Let a route solution be denoted $r = (r_1, ..., r_k, ..., r_m)$, where r_k is an ordered node sequence representing agent A_k's route. A solution r is MAOPCC feasible if there is a time sequence in visiting the route nodes in each r_k one after another such that each agent A_k can satisfy the set of constraints in (10). A naive yet exhaustive approach for optimizing MAOPCC is to enumerate and verify the MAOPCC feasibility of all route solutions. However, for each agent with n nodes (including starting and ending nodes), there exist $(n-2)!$ possible routes. As such, it has to enumerate a total number of $(n-2)!m$ possible route solutions, which increases rapidly with the number of nodes and number of agents. To handle this issue, we first proposed an exact method using the technique of branch and bound to speed up the search by pruning the route solutions that are not promising.

A. Branch and Bound (BB)

Starting with null route assignments for all agents, the branch and bound algorithm branches on the route assignments for a particular agent r_p $(1 \leq p \leq m)$ and recursively solves the remaining sub-problem given the route assignments for the previous p agents. The BB algorithm will backtracking if solving the sub-problem cannot produce a better route solution than the currently found best route solution r^{best} that is MAOPCC feasible. To realize this, an upper bound of the sub-problem is required and it is obtained by solving its relaxed problem given as follows.

Let the function $f(r_k)$ sum up the rewards of all the route nodes for r_k, i.e.,

$$f(r_k) = \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ik} x_{ijk}^{t}. \quad (11)$$

$$\max \sum_{k=p+1}^{m} f(r_k) \quad (12)$$

subject to $\forall k \in \{p+1, ..., m\}, d \in \{2, ..., n-1\}$:

$$\sum_{t=1}^{T}\sum_{j=1}^{n} x_{1jk}^{t} = \sum_{t=1}^{T}\sum_{i=1}^{n} x_{ink}^{t} = 1 \quad (13)$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n} x_{idk}^{t} = \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \text{ and } \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \leq 1 \quad (14)$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n}(t + t_{id} + e_d)x_{idk}^{t} = \sum_{t=1}^{T} t(\sum_{j=1}^{n} x_{djk}^{t}) \quad (15)$$

$$\sum_{t=1}^{T}\sum_{j=1}^{n} t * x_{1jk}^{t} = T_k^1 \text{ and } \sum_{t=1}^{T}\sum_{i=1}^{n} t * x_{ink}^{t} \leq T_k^n \quad (16)$$

Assume R_k includes all possible candidate routes for the agent A_k that start its trip from node v_1 at the time step T_k^1 and ends its trip at the node v_n before the time step T_k^n, then the upper bound g of the sub-problem in (**??**)-(**??**) is estimated according to the function given by

$$g = \sum_{k=1}^{p} f(r_k) + \sum_{k=p+1}^{m} max_{r_k \in R_k} f(r_k) \quad (17)$$

The BB algorithm terminates when all the branches have been explored. The best solution r^{best} is initially set to $\emptyset$ and it will be updated once a better MAOPCC feasible solution is

found. However, to improve BB's computational efficiency, a good solution r^{best} can be initialized by other heuristics such as the Sequentail Algorithm (SA) presented below.

B. Sequential Algorithm (SA)

Though BB can prune the solution space quite substantially, it is still limited by its systematical exploration of all branches. In addition, the computation of the upper bound of g in (12)-(16) requires solving a set of computationally hard OP problem instances, which has a complexity of $O(m(n-2)!)$. As a result, the computational time of BB still increases rapidly with increasing number of nodes and agent count. To overcome the computation issue, we introduce a sequential approach next so as to find a good MAOPCC solution more quickly.

The Sequential Algorithm decouples the MAOPCC problem into m sub-problems given by:

$$\max \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ik}x_{ijk}^{t} \ (1 \le k \le m) \tag{18}$$

subject to $\forall d \in \{2, ..., n-1\}$:

$$\sum_{t=1}^{T}\sum_{j=1}^{n} x_{1jk}^{t} = \sum_{t=1}^{T}\sum_{i=1}^{n} x_{ink}^{t} = 1 \tag{19}$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n} x_{idk}^{t} = \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \text{ and } \sum_{t=1}^{T}\sum_{j=1}^{n} x_{djk}^{t} \le 1 \tag{20}$$

$$\sum_{t=1}^{T}\sum_{i=1}^{n}(t + t_{id} + w_{dk}^{t+t_{id}} + e_d)x_{idk}^{t} = \sum_{t=1}^{T} t(\sum_{j=1}^{n} x_{djk}^{t}) \tag{21}$$

$$\sum_{t=1}^{T}\sum_{j=1}^{n} t * x_{1jk}^{t} = T_k^1 \text{ and } \sum_{t=1}^{T}\sum_{i=1}^{n} t * x_{ink}^{t} \le T_k^n \tag{22}$$

$\forall d \in \{2, ..., n-1\}, t \in \{1, ..., T\}$:

$$q_{dk}^{t} = q_{dk}^{t-1} - \sum_{a=1}^{k-1}\sum_{j=1}^{n} x_{dja}^{t} + \sum_{i=1}^{n} x_{idk}^{t-t_{id}} \sum_{a=1}^{k-1}\sum_{i=1}^{n} x_{ida}^{t-t_{id}} \tag{23}$$

$$x_{iik}^{t} = 0 \text{ and } x_{ijk}^{t}, z_{ik}^{t} \in \{0,1\}, y_{ik}^{t} \in Z \text{ for } \forall i, j \in \{1, ..., n\} \tag{24}$$

and then solves each sub-problem in a sequential manner, starting from $k = 1$ until $k = m$. Given the obtained solutions for sub-problems 1 to $k-1$, a subsequent sub-problem k is solved by employing a greedy approach, which repeatedly inserts an unvisited attraction node into the feasible position of route r_k until no feasible insertion is available. To find the best insertion position for each unvisited attraction node v_d in the route r_k, the minimal traversing time incurred by inserting this node into the feasible position of the route is determined, which is recorded as Δ_{dk}^{min} ($\Delta_{dk} = t_{id} + e_d + t_{dj} - t_{ij}$ if node v_d is inserted between the nodes v_i and v_j). To be a feasible insertion position, the new route produced for agent A_k should satisfy the constraints in (12). Note that Δ_{dk}^{min} of the unvisited attraction node will be set to ∞ if it has no feasible insertion position. To select the unvisited attraction node to be inserted, $(s_{dk})^2/\Delta_{dk}^{min}$ is calculated for each unvisited attraction node v_i and the attraction node with the highest $(s_{dk})^2/\Delta_{dk}^{min}$ is selected. In each sub-problem, a number of $(n-2)(n-1)^2/4$ candidate routes has to be verified for MAOPCC feasibility in the worst case. As such, SA has a polynomial complexity of $O(m * n^3)$ in verifying the MAOPCC feasibility of route solutions.

C. Probabilistic Iterated Local Search (PILS)

The Sequential Algorithm decouples the MAOPCC problem into several sub-problems with lower dimensional search space and therefore is able to find a MAOPCC solution quickly. However, the computational efficiency of SA sacrifices the optimality of the solution. To strengthen the heuristic, we apply a probabilistic version of the standard iterated local search (PILS). As in a standard ILS, our proposed PILS algorithm uses an insertion and an exchange operator to find a locally optimal solution, and then perturbs the solution so as to escape from it. Unlike the stardard ILS however, we introduce two additional parameters (β and ρ below) that control the search probabilistically, in terms of the adding and removing attractions nodes from routes. PILS terminates when the current best solution has not been improved after δ iterations.

The local search procedure finds a locally optimal solution by interactively applying an insertion move in line 8 and an exchange move in line 11 until no further improvement is possible. The insertion move repeats to probabilistically add an unvisited attraction node of one of the agents into the best feasible position of its route until no feasible insertion is possible. For each unvisited attraction node of all agents, it is still determined by Δ_{dk}^{min} described in section IV-B, however, the selection of an unvisited attraction node to be inserted is probabilistically chosen, and the selection probability distribution is directly proportional to their costs of $\beta * (s_{dk})^2/\Delta_{dk}^{min}$.

The exchange move replaces a visited attraction node in an agent route with one of its unvisited attraction nodes until no feasible exchange move can be made. Same with insertion move, the new route produced by exchange moves is constrained within its time budget and the selected unvisited attraction node v_i is added into the feasible position of the route r_k with Δ_{ik}^{min}. Different with insertion move, the visited attraction node with the lowest score is selected to be replaced while only the unvisited attraction node with a higher reward than the selected visited node is accepted in the exchange move.

To escape from a local optimum, the solution is perturbed by removing a number of q_k number of attraction nodes from each route r_k. The integer value of q_k (=0, 1, 2,..., α_k, where α_k is the maximum number of attraction nodes in the route r_k) is generated based on the probability function given by

$$P_{\alpha_k,\rho}(q_k) = \begin{cases} (1-\rho) * ((\alpha_k - q_k + 1)/(\sum_{i=0}^{\alpha_k}(i+1))) & \text{if } \rho < 1 \\ 1/(\alpha_k + 1) & \text{if } \rho = 1 \end{cases} \tag{25}$$

Note that as the parameter ρ increases, the probability of q_k with a lower value decreases while the probability of q_k with a higher value increases. In so doing, the perturbation strength gradually increases when no improving solution is found as the number of iteration increases. Given a non-uniform value of q_k for each route r_k, the attraction nodes are removed based on a uniform distribution.

V. Experimental Results

The performance of our proposed algorithms are evaluated and compared on two sets of test instances with uniformly randomly generated positions and rewards (ranging from 1-20) for each node and each agent. A set of small instances with 12 nodes and the agent count ranging from 5 to 20 in a 50x50 grid map are used to compare their performance. Here, the capacity of each attraction node is set to 2. Another larger set of instances with 22 nodes and agent number varying from 20 to 100 in a larger 100x100 grid map is used to compare performance of competing algorithms when instances are larger. Here, the node capacity in this set is set to 5. In both sets of instances, time budgets of all agents are set to 100 time units and all agents start their trips at different times ranging from 0 to m. Without loss of generality, service times of all nodes are set to 1 time unit. The performance is mainly evaluated based on two different criteria. The first is total scores, which is defined as the average total reward of all agents in 10 random instances. The second is running time, which is defined as the average computational time taken to compute the solution in 10 random instances. In addition, an optimality gap and a relative gap are defined as the percentage of the average reward loss of the solution obtained compared to the optimal solution and an upper bound of the solution respectively in 10 random instances, and they are employed here to investigate the performance of the comparative algorithms. The upper bound of the solution for each MAOPCC instance is obtained by solving OP problems for each agent.

The simulation program is coded in C++ programming language and executed on an Intel Xeon CPU X7542 2.67GHz 24 cores processors with 128GB RAM.

The following experimental results compare the proposed three approaches on a set of small instances. SA decomposes MAOPCC problem into several orienteering sub-problems, and then solves each sub-problem in a sequential manner. Each subsequent orienteering sub-problem is solved by employing the insertion move proposed in [14] with respect to the previously obtained sub-solutions. To speed up the BB algorithm, the solution obtained by SA is set as the initial best solution r^{best}. PILS will be terminated if no improvement is obtained after δ iterations. For our experiments, we set $\delta = 80$.

For smaller instances, the BB algorithm, specifically designed to find an optimal solution for MAOPCC, is able to provide the best total scores performance over other competing algorithms. However, due to its rapidly increasing search space, its computational time increases rapidly as number of agents increases (see the results in Table II). By solving MAOPCC in a decoupled way, SA is very fast in finding a good MAOPCC solution. But it has a larger optimality gap compared with PILS. From $m = 5$ to $m = 20$, the optimality gap of SA increases from 2.96% to 6.1%. This translates that the performance of SA in finding a good MAOPCC solution could deteriorate as the problem size increases. In contrast, the optimality gap of PILS ($\delta = 80$) is 0.79% when m=5 while 2.13% when $m = 20$. Moreover, the solution quality of PILS can be further improved if δ is set to larger values (see Table III). However, due to its probabilistic and heuristic nature, the improvement of solution quality in PILS comes at the expensive of a higher computational time.

Total Scores	**n=12**			
	m=5	**m=10**	**m=15**	**m=20**
BB+SA	357.7	763.9	1109.7	1379.9
SA	347.6 (2.96%)	748.1 (2.17%)	1060.5 (4.8%)	1298.1 (6.1%)
PILS($\delta = 80$)	354.8 (0.79%)	756.9 (0.93%)	1094.9 (1.52%)	1351.2 (2.13%)

TABLE I
AVERAGE TOTAL REWARDS OF ALL COMPETING ALGORITHMS IN 10 RANDOM SCENARIOS WITH 12 NODES AND AGENT NUMBER RANGING FROM 5 TO 20. THE OPTIMALITY GAP IS GIVEN IN BRACKETS.

Since the proposed PILS algorithm is developed based on the work in [14], the following experiments will compare our proposed PILS algorithm with the ILS algorithm proposed in [14]. However, since ILS is originally proposed to solve TOPTW, to apply ILS for MAOPCC, its original insertion move for single agent is replaced with our proposed insertion move for multiple agents. Apart from the ILS algorithm, SA was also compared here. The non-improvement iteration bounds of $\delta = 80$ was used for both ILS and PILS.

Running Time (s)	**n=12**			
	m=5	**m=10**	**m=15**	**m=20**
BB+SA	2.0	36.0	2210.0	25016.0
SA	0.002	0.005	0.005	0.007
PILS($\delta = 80$)	1.464	11.7	35.8	98.5

TABLE II
AVERAGE COMPUTATIONAL TIME OF ALL COMPETING ALGORITHMS IN 10 RANDOM SCENARIOS WITH 12 NODES AND THE AGENT COUNT RANGING FROM 5 TO 20.

PILS	$\delta = 80$	$\delta = 500$	$\delta = 2000$	$\delta = 10000$
Total Scores	1351.2	1356.7	1362.5	1362.5
Running Time (s)	98.5	444.8	1992.0	8801.7

TABLE III
SENSITIVITY ANALYSIS OF THE PERFORMANCE OF PILS IN 10 RANDOM SCENARIOS WITH 12 NODES AND 20 AGENTS AGAINST THE ALLOWABLE ITERATION BOUND δ.

For the larger instances, SA still exhibits its superior performance in speed and solution quality (see Tables IV and V). Though ILS is able to produce a better MAOPCC solution, its solution quality cannot be further improved as the non-improvement iteration bound increases, due to its monotonic remove strategy in the perturbation procedure in which a uniform number of consecutive attraction nodes are deleted from

Total Scores	n=22				
	m=20	m=40	m=60	m=80	m=100
PILS($\delta = 80$)	834.3 (3.19%)	2094.3 (1.92%)	2407.3 (1.46%)	3826.3 (3.22%)	4332.4 (4.31%)
ILS($\delta = 80$)	813.3 (5.40%)	2018.8 (5.50%)	2403.0 (1.75%)	3751.6 (5.03%)	4216.7 (6.86%)
SA	805.4 (6.23%)	1973.8 (7.33%)	2372.5 (2.72%)	3713.4 (6.00%)	4180.4 (7.60%)

TABLE IV
AVERAGE TOTAL REWARDS OF ALL COMPETING ALGORITHMS IN 10 RANDOM SCENARIOS WITH 22 NODES AND THE AGENT COUNT RANGING FROM 20 TO 100. RELATIVE GAP IS INDICATED IN BRACKETS.

the route of each agent. In the worst case, ILS could be stuck in a cyclic search for a set of solutions. In contrast, our proposed PILS algorithm employs a probabilistic remove strategy, which allows the diverse removal of the visited attraction nodes and the non-uniform number of attraction nodes to be removed in order to take into account the difference among agents like their heterogeneous preference on different attractions and therefore achieve the coordination between the routes of all agents. The experimental results in Table IV indeed show that this consideration contributes to the further improvement of the solution quality in PILS. Moreover, PILS is able to produce a better solution than ILS at the expense of a lower computational time (see Table V).

We have done five independent runs for each algorithm per instance. According to paired t-test, we can be 95% sure that PILS is statistically significantly better than SA and ILS.

Running Time (s)	n=22				
	m=20	m=40	m=60	m=80	m=100
PILS($\delta = 80$)	28.3	403.3	839.6	2217.6	5345.1
ILS($\delta = 80$)	54.2	766.9	1641.4	4915.6	8000.0
SA	0.005	0.014	0.019	0.025	0.027

TABLE V
AVERAGE COMPUTATIONAL TIME OF ALL COMPETING ALGORITHMS IN 10 RANDOM SCENARIOS WITH 22 NODES AND THE AGENT COUNT RANGING FROM 20 TO 100.

Finally, the following experiment evaluates the performance of the SA and PILS algorithms for some specific hard instances in the 100x100 grid map, in which the scores of two of the attraction nodes (v_{11} and v_{21}) are specifically set to a value of 10 for agents $A_1 - A_{m/2}$ and 20 for agents $A_{m/2+1} - A_m$ while others are uniformly set to 1 for all agents, the starting time of agents $A_1 - A_m$ is randomly chosen between the time interval 0 to 20, in addition, the capacity and service time of the attraction nodes v_{11} and v_{21} are changed to 1 and 5 while others are still fixed to 5 and 1. The instances are challenging as they require fine coordination of priorities in visiting the attraction nodes between agents.

Being a decoupled approach, SA solves each sub-problem for each agent in a sequential manner and plans the route for one agent at a time. This may lead to the issue that the previously obtained sub-solution may compromise good sub-solutions for subsequent sub-problems. In these challenging instances, due to the planning sequence from A_1 to A_m, the

Total Scores	n=22				
	m=20	m=40	m=60	m=80	m=100
SA	352.9 (46.1%)	459.3 (65.7%)	510.3 (74.6%)	513.9 (80.3%)	552.2 (82.9%)
PILS($\delta = 80$)	566.2 (13.4%)	708.8 (46.9%)	845.3 (57.8%)	863.6 (66.8%)	933.8 (71.0%)

TABLE VI
AVERAGE TOTAL REWARDS OF THE SA AND PILS ALGORITHMS IN 10 HARD RANDOM SCENARIOS WITH 22 NODES AND THE AGENT COUNT RANGING FROM 20 TO 100. RELATIVE GAP IS INDICATED IN BRACKETS.

Total Scores	n=22				
	m=20	m=40	m=60	m=80	m=100
SA	0.022	0.051	0.070	0.064	0.072
PILS($\delta = 80$)	585.5	2946.3	15595.7	36309.8	69850.3

TABLE VII
AVERAGE COMPUTATIONAL TIME OF THE SA AND PILS ALGORITHMS IN 10 HARD RANDOM SCENARIOS WITH 22 NODES AND THE AGENT COUNT RANGING FROM 20 TO 100.

previous agents $A_1 - A_{m/2}$ may occupy the nodes v_{11} and v_{21} and therefore prevent agents $A_{m/2+1} - A_m$ visit v_{11} and v_{21} due to the limited capacity. Instead of considering one agent at a time, PILS plans the routes for all agents simultaneously. It was observed that PILS is able to find a MAOPCC solution that is superior to SA.

In summary, we have introduced three approaches to solve the computationally hard MAOPCC problem for different purposes. When the problem size is small and solution quality is a primary concern, the proposed BB algorithm can be used to find an optimal MAOPCC solution in an acceptable time. As the problem size becomes larger, our proposed heuristic PILS algorithm can be used to find a good solution in a reasonable time. When the running time is more critical, SA could be a good choice.

VI. CONCLUSION

In this paper, we introduce a multi-agent variant of the orienteering problem, namely multi-agent orienteering problem with capacity constraints (MAOPCC), in which multiple agents concurrently visit the same set of nodes with a capacity constraint for each node. Such capacity constraints limit the number of agents who can be served concurrently. Our objective is to find routing sequences for all agents so as to maximize the sum of collected rewards from all agents. Besides the exponential growth in problem size, MAOPCC also faces the modeling difficulty of having to explicitly model the queueing of agents at crowded nodes.

To address this problem, we propose an optimal branch and bound approach (BB), a computationally fast sequential algorithm (SA) and a probabilistic iterative local search heuristic (PILS). The PILS heuristic employs the technique of iterated local search that systematically searches in a combinatorial route space R which is a Cartesian product of m OP search spaces. For each solution r in the route space R, we provide an approach that is able to model the exact queuing sequence between agents at an attraction node given r and therefore successfully transforms a route solution r to

a MAOPCC solution. Though the performance of BB can be improved by using initial solutions, it is undeniable that it still suffers from memory and computational time issues due to its systematic search in the combinatorial route space. Thus, BB is appropriate for solving small-scale MAOPCC problems. SA is computationally very fast in finding a feasible MAOPCC solution, but it sacrifices the optimality of solution and is not desirable for MAOPCC scenarios where the solution quality is one of the primary concerns. In contrast, at each iteration, our PILS algorithm searches for a locally optimal solution in a small local neighborhood by using computationally efficient insertion and exchange moves, and then tries to find an unexplored and promising local neighborhood by employing a probabilistic remove operation that favors the coordination of the routes for all agents. By setting an appropriate iteration bound, PILS could strike a good tradeoff between the solution quality and the computational time incurred.

ACKNOWLEDGMENT

This research is jointly funded by the National Research Foundation Singapore under its Corp Lab@University scheme and its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

REFERENCES

[1] A. Gunawan, H. C. Lau and P. Vansteenwegen, Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research, 255(2):315–332, (2016).

[2] C. Chen, S-F. Cheng and H.C. Lau. Multi-agent orienteering problem with time-dependent capacity constraints. Web Intelligence and Agent Systems Journal, 12:347-358,(2013).

[3] B.L. Golden, E.A. Wasil and I-M. Chao, Theory and methodology for the team orienteering problem. European Journal of Operational Research, 88:464-474,(1996).

[4] G. Laporte and S. Martello, The selective travelling salesman problem. Discrete Applied Mathematics. 26:193-207,(1990).

[5] M. Fischetti, J. Salazar, P. Toth, Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing. 10:133-148,(1998).

[6] B. Golden, L. Levy and R. Vohra, The orienteering problem. Naval Research Logistics. 34:307-318,(1987).

[7] I-M. Chao, B. L. Golden, E. A. Wasil, Theory and methodology. A fast and effective heuristic for the orienteering problem. European Journal of Operational Research. 88: 475-489,(1996).

[8] V. Campos, R. Marti, J. Sa ?nchez-Oro and A. Duarte, GRASP with path relinking for the orienteering problem. Journal of the Operational Research Society. 65:1800-1813,(2014).

[9] M. G. Kantor and M. B. Rosenwein, The orienteering problem with time windows. Journal of Operational Research Society, Vol.43, No.6, pp.629-635,(1992).

[10] G. Righini and M. Salani, Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming. Computer & Operations Research, 36:1191-1203,(2009).

[11] G. Righini and M. Salani, New dynamic programming algorithms for the resource constrained elementary shortest path. Networks 51(3), 155-170,(2008).

[12] I-M. Chao, B. L. Golden, E. A. Wasil, Theory and Methodology. The team orienteering problem. European Journal of Operational Research, 88:464-474,(1996).

[13] S. Boussier, D. Feillet and M. Gendreau, An exact algorithm for team orienteering problems. 4OR, Vol.5, issue 3, pp.211-230,(2007).

[14] P. Vansteenwegen, W. Souffriau, G.V. Berghe and D.V. Oudheusden, Iterated local search for the team orienteering problem with time windows, Computer & Operations Research, 36: 3281-3290,(2009).