Computers and Operations Research 89 (2018) 1–12

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Review

Metaheuristics for the tabu clustered traveling salesman problem

Tianjiao Zhang[a,b,c,*], Liangjun Ke[a,b,c,*], Jing Li[a,b,c], Jisheng Li[a,b,c], Jingqi Huang[a,b,c], Zexi Li[a,b,c]

[a] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China
[b] State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
[c] State Key Laboratory of Astronautic Dynamics, Xi'an Satellite Control Center, Xi'an, China

ARTICLE INFO

Article history:
Received 9 February 2016
Revised 23 March 2017
Accepted 11 July 2017
Available online 13 July 2017

Keywords:
Tabu clustered traveling salesman problem
TT&C resources scheduling problem
Ant Colony Optimization
Greedy Randomized Adaptive Search Procedure
Metaheuristics

ABSTRACT

This paper considers a new variant of traveling salesman problem (TSP), called tabu clustered TSP (TCTSP). The nodes in TCTSP are partitioned into two kinds of subsets: clusters and tabu node sets, then the salesman has to visit exactly one node for each tabu node set and ensures that the nodes within a same cluster are visited consecutively, and the problem calls for a minimum cost cycle. The TCTSP can be used to model a class of telemetry tracking and command (TT&C) resources scheduling problem (TTCRSP), the goal of which is to efficiently schedule the TT&C resources in order to enable the satellites to be operated normally in their designed orbits. To solve it, two metaheuristics combined with path relinking are proposed. The one is Ant Colony Optimization (ACO) and the other is Greedy Randomized Adaptive Search Procedure (GRASP). The proposed algorithms are tested on the benchmark instances and real-life instances of the TTCRSP. The computational results show that the hybrid ACO with two path relinking strategies works the best among the studied metaheuristics in terms of solution quality within the same computational time.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied combinatorial optimization problems. It aims at finding the shortest tour that visits the nodes in a given weighted complete graph exactly once and returns to the origin node. A widely studied variant of the TSP is the Clustered TSP (CTSP) (Gutin and Punnen, 2002), which was firstly studied in Chisman (1975). In the CTSP, cities within the same cluster must be visited consecutively. The CTSP can be used to model a broad range of real-life applications, such as vehicle routing (Laporte et al., 2002), manufacturing (Lokin, 1979), warehousing problem (Chisman, 1975), computer disk defragmentation, production planning and several kinds of scheduling problems (Laporte et al., 2002).

Since the CTSP is an NP-hard problem, exact algorithms can merely find an optimal solution for very limited-size instances due to finite computational resource. Instead, many researchers focused on finding a satisfactory solution within reasonable time. Jongens and Volgenant (1985) proposed a Lagrangean relaxation algorithm to obtain lower bounds of the optimal tour lengths for a set of CTSP test instances with the number of vertices varying from 80 to 150 and different sizes of clusters. Gendreau et al. (1994) first transformed the CTSP into a TSP and then applied the GENIUS heuristic procedure. Laporte et al. (1997) proposed a tabu search heuristic to solve the CTSP with a prespecified order of visiting the clusters. In Potvin and Guertin (1996), 1998), genetic algorithm (GA) was applied on the CTSP. Compared with the GENIUS heuristic (Gendreau et al., 1994) and the lower bounds obtained in Jongens and Volgenant (1985), GA can obtain results within 5.5% of the lower bound. Mestria et al. (2013) employed the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic.

In this paper, a new variant of the CTSP, motivated by a real-life application of China Satellite Control Center (CSCC), arises when imposing the additional constraint that a feasible tour is allowed to visit only a subset of the nodes. Every day, CSCC has to coordinate communications between satellites and ground stations so as to keep the orbiting satellites working effectively. Each satellite needs services in a specific time window among possible alternative slots and generally asks for 4–6 times one day. The problem of scheduling all the services is referred as the TT&C resources scheduling problem (TTCRSP) (Zhang et al., 2014), which aims to allocate resources (i.e., contact opportunity windows) to as many services as possible during scheduling period (Bianchessi et al., 2007; Karapetyan et al., 2015; Marinelli et al., 2011; Wu et al., 2013). Since the number of satellites grows year by year, more services ask for certain resources than can be accommodated. Therefore the TTCRSP is an oversubscribed problem (Barbulescu et al., 2004).

* Corresponding authors.
E-mail addresses: tiantian880407@163.com, tiantian8847@aliyun.com (T. Zhang), keljxjtu@xjtu.edu.cn (L. Ke), carol_lee_0727@sina.com (J. Li), lijisheng@cashq.ac.cn (J. Li), 397790194@qq.com (J. Huang), zexi57@163.com (Z. Li).

http://dx.doi.org/10.1016/j.cor.2017.07.008
0305-0548/

The TTCRSP is closely related to the CTSP. The satellite contact opportunity window with a scheduling benefit in TTCRSP can be viewed as a node in TSP. When two contact opportunity windows can be assigned to their corresponding services consecutively, there is an edge between them with an associated cost. Under such a situation, satellite control center would like to design a sequence of the chosen contact opportunity windows so as to complete all the services and maximize the total profit. In order to reduce the sensor opening and slewing time for less energy consumption, some services are generally grouped into clusters so that they can be completed together. More specially, each service has a set of alternative contact opportunity windows, but exactly one is chosen according to the domain constraints. The contact opportunity windows belonging to a same service are mutually exclusive, and node sets consisting of mutual exclusive nodes are tabu node sets in which only one node can be visited. Although the CTSP has been generalized in the literature (Gutin and Punnen, 2002) by taking into account some restrictions on the nodes in a cluster, to the best of our knowledge, our problem does not belong to any existing variant of the TSP. Due to the existence of tabu node sets, we named this new variant of the problem Tabu CTSP (TCTSP).

To deal with the TCTSP, a metaheuristic which combines Ant Colony Optimization (ACO) with path relinking is proposed. ACO is a swarm intelligence optimization algorithm which has been successfully used to solve many combinatorial optimization problems, such as TSP (Dorigo et al., 1996; Feo et al., 1994), while path relinking is an effective intensification search mechanism. Furthermore, we also extend the hybrid GRASP in Mestria et al. (2013) to solve the TCTSP.

The rest of this paper is organized as follows. Section 2 describes the formulation of the TCTSP. Section 3 presents the ACO heuristics proposed for the TCTSP. Section 4 gives simulation experiments and performance analysis. Section 5 studies a real-life case of the TCTSP. Finally, Section 6 concludes the main results.

2. A description of the TCTSP

The TCTSP is defined on a complete (loop - free) undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes and $E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j)$ is the set of edges. In this work, the Euclidean distance c_{ij} between v_i and v_j is set as the cost of each edge $(v_i, v_j) \in E$ and the edge costs satisfy the triangle inequality. In addition, the nodes are properly partitioned into two kinds of subsets : *clusters* $V_1, \cdots, V_l$ and *tabu node sets* $T_1, \cdots, T_m$. A cycle is called *feasible* if it visits the nodes in each cluster consecutively and contains exactly one node from each tabu set. TCTSP then consists of finding a feasible cycle $T \subset E$ whose cost $\sum_{(v_i,v_j)\in T} c_{ij}$ is minimized. When there is only one cluster and each tabu node set contains only one node, TCTSP is reduced to TSP. The problem involves two related decisions:

- choose a node subset $S \subseteq V$, such that $|S \cap V_h| = 1$ for all $h = 1, \cdots, l$;
- find a minimum cost Hamiltonian cycle in the subgraph of G introduced by S.

Fig. 1 shows a feasible solution to an instance of the TCTSP that has three clusters and ten tabu node sets. The solution is a Hamiltonian tour such that the nodes of each cluster are visited consecutively and exactly one node in each tabu node set is visited. The nodes 1,4,11 and 19 in a tabu set may belong to different clusters, the dotted-line edges (10,11),(2,13) and (14,17) show the inter-cluster connections and the full line ones are the intra-cluster connections.

The binary decision variable $x_{ij} = 1$ if and only if the salesman proceeds from city v_i to city v_j. The formulation of the TCTSP can

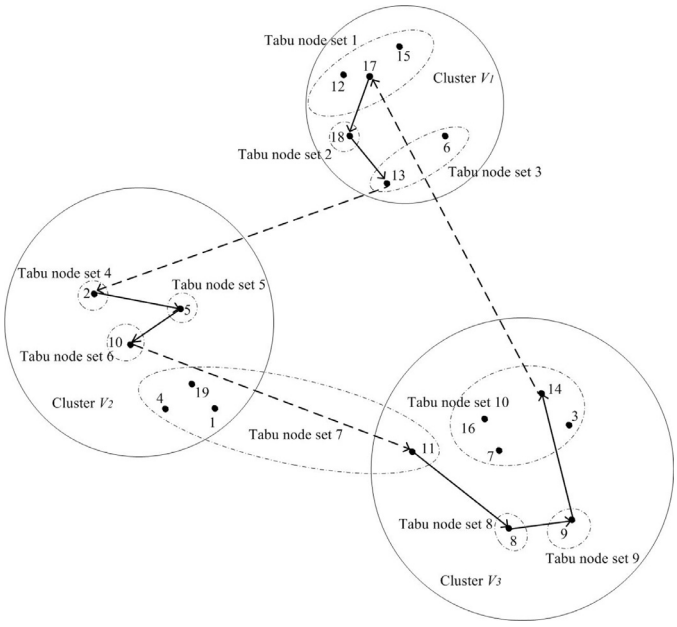

Fig. 1. A feasible solution to an instance of the TCTSP.

be described as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} \leq 1, \forall i \in V \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1, \forall j \in V \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j\in T_k} x_{ij} = 1, \forall T_k \subset V, |T_k| \geq 1, k = 1, \cdots, m, \forall i \in V \tag{4}$$

$$\sum_{i\in T_k}\sum_{j=1}^{n} x_{ij} = 1, \forall T_k \subset V, |T_k| \geq 1, k = 1, \cdots, m, \forall j \in V \tag{5}$$

$$\sum_{i\notin V_k}\sum_{j\in V_k} x_{ij} = 1, \forall V_k \subset V, |V_k| \geq 1, k = 1, \cdots, l \tag{6}$$

$$\sum_{i\in V_k}\sum_{j\notin V_k} x_{ij} = 1, \forall V_k \subset V, |V_k| \geq 1, k = 1, \cdots, l \tag{7}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V \tag{8}$$

The goal is to minimize the total distance traveled by the salesman. Constraints (2) and (3) mean that each city can be visited at most once. Constraints (4) and (5) state that exactly one city of each tabu node set can be visited. Constraints (6) and (7) ensure that cities within the same cluster must be visited consecutively. Constraint (8) defines the domain of every decision variable.

3. The proposed ACO heuristics for the TCTSP

ACO is a metaheuristic framework for solving discrete combinatorial optimization problems. It takes inspiration from the foraging behavior of ant species (Dorigo and Birattari, 2010). ACO represents

an optimization problem by a construction graph and uses N artificial ants to walk on the graph where N is the size of antcolony. Each ant constructs solutions iteratively and its behavior is guided by pheromone and heuristic information. The main iterative procedure of ACO consists of three steps. At the first step, each ant builds a solution according to the transition rule. Subsequently a local search procedure is employed to improve every constructed solution. At the last step, pheromone is updated. The iterative process terminates when a stopping criterion is satisfied. The main procedure of ACO metaheuristic is given in Algorithm 1.

Algorithm 1 The Ant Colony Optimization metaheuristic.

```
Set parameters, initialize pheromone trails
while The stopping criterion is not met do
    Construct_Ant_Solutions( )
    Local_Search( )
    Update_Pheromones( )
end while
```

3.1. The construction phase

At first, a penalization is added to all inter-clusters edges such that the TCTSP is transformed to the Tabu Traveling Salesman Problem (TTSP). The TTSP is a TSP, each vertex belongs to a tabu node set. Then the solution construction procedure of the ACO algorithm is applied to solve the TTSP. Algorithm 2 describes the construction phase. The N artificial ants construct solutions in the same manner: starting from an empty solution $s_p = \emptyset$, at each construction step, the partial solution s_p is extended by adding a city as a solution component from the feasible city set $N(s_p) \subset V$ according to the random-proportional rule:

Algorithm 2 Construct_Ant_Solutions().

```
Penalize_Edges( )
for (k = 1; k ≤ N; k++) do
    choose a city i randomly from the city set V as the start city
    s_p^k ← i
    N(s_p^k) ← V \ {i}
    while N(s_p^k) ≠ ∅ do
        choose a city j ∈ N(s_p^k) with probability p_ij^k
        s_p^k ← Add_City(s_p^k, j)
        T_j ← the tabu node set of city j
        for (q = 1; q ≤ |T_j|; q++) do
            if T_j[q] ∈ N(s_p^k) then
                N(s_p^k) ← N(s_p^k) \ {T_j[q]}
            end if
        end for
    end while
end for
return s_p^k
```

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^a \cdot \eta_{ij}^b}{\sum_{l \in N(s_p^k)} \tau_{il}^a \cdot \eta_{il}^b}, & \text{if city } l \in N(s_p^k) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $N(s_p)$ is defined as the set of cities that can be added to the partial solution without violating any constraints (2)–(8). τ_{ij} is the pheromone of edge (i, j). The heuristic information of edge (i, j) is chosen as $\eta_{ij} = \frac{1}{c_{ij}}$. a and b are the parameters that control their relative importance.

Once an ant has chosen a city to extend the partial solution, the feasible city set is modified by removing the chosen city and some forbidden cities which are in the same tabu node set as the chosen city. If $N(s_p)$ is empty, the ant completes its tour.

3.2. Local search phase

Once an ant has finished its solution, the solution will be improved by local search. The local search phase proposed for the TCTSP uses the *2-Opt* operator to exchange intra-cluster and inter-cluster edges. Algorithm 3 shows the local search procedure. The input is an initial solution s with m edges and the output is the improved solution s^*. At each step, two nonadjacent edges $s(v_i, v_{i+1})$ and $s(v_j, v_{j+1})$ are selected first. If both edges are inter-cluster or intra-cluster edges, then the *2-Opt* operator performs a move on these two edges to obtain a new intermediate solution s', if a new solution with better cost is found, the incumbent solution is replaced and the local search procedure repeats until no better solution can be found.

Algorithm 3 Local_Search().

```
s* ← s
improve ← true
while improve ← false do
    for (i = 1; i ≤ m − 1; i++) do
        for (j = l + 1; j ≤ m; j++) do
            if s(v_i, v_{i+1}) is not adjacent to s(v_j, v_{j+1}) then
                if s(v_i, v_{i+1}) and s(v_j, v_{j+1}) are on the same cluster
                or inter-cluster edges then
                    s' ← 2_Opt(s(v_i, v_{i+1}), s(v_j, v_{j+1}))
                    if f(s') < f(s*) then
                        s* ← s'
                    end if
                end if
            end if
        end for
    end for
end while
return s*
```

3.3. Pheromone update

After each constructed solution has been improved by local search, ants update the pheromone matrix according to the pheromone update rule of MAX-MIN ant system (Stützle and Hoos, 2000): pheromone evaporates uniformly on all edges in order to allow ants to forget partial historic memory, then the ant which has constructed the shortest tour in this cycle deposits a fraction of pheromone on the edges of the shortest tour, finally the pheromone bound is imposed. The pheromone τ_{ij} of edge (i, j) is updated as follows:

$$\begin{aligned} &\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \sum_{(i,j) \in A_k} \Delta\tau_{ij} \\ &\text{if} \tau_{ij} < \tau_{min}, \text{then } \tau_{ij} \leftarrow \tau_{min} \\ &\text{if} \tau_{ij} > \tau_{max}, \text{then } \tau_{ij} \leftarrow \tau_{max} \end{aligned} \tag{10}$$

where τ_{max} and τ_{min} are the upper and lower bounds of pheromone, respectively. Parameter ρ $(0 \le \rho \le 1)$ is the evaporation parameter. A_k is the shortest tour which may be the best-so-far or iteration-best, and $\Delta\tau_{ij}$ is the quantity of pheromone deposited on edge (i, j) by the best ant which is defined as follows:

$$\Delta\tau_{ij} = \begin{cases} \dfrac{1}{cost(A_k)}, & \text{if}(i,j) \in A_k \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

3.4. ACO with path relinking strategies

Path relinking has been widely used to enhance the intensification search of metaheuristic (Reghioui et al., 2007). It was originally introduced by Glover (1997) to improve solutions obtained by tabu search. Since high-quality solutions often share a significant portion of attributes, path relinking explores the trajectories linking high-quality solutions in search space so as to find a better one. Therefore, in this section, we redefined some components of the path relinking strategy according to the properties of the TCTSP and combined it with ACO algorithm.

In path relinking, between two high-quality solutions, the initial solution is set as the solution with the better cost and the target solution is the other one. Path relinking works as follows: at first, the distance between the initial solution and the target solution is computed. Then paths between these two solutions are explored and generated by applying neighborhood moves to the initial solution, the procedure terminates until the initial solution is completely transformed into the target solution. The following subsections describe how to compute the distance and show the details of trajectory exploration, finally present how to combine path relinking with ACO.

3.4.1. Distance measure

The distance $\Delta(s_{ini}, s_{tar})$ for the TCTSP is the numbers of tabu node set pairs of consecutive elements in the initial solution s_{ini} that are broken in the target solution s_{tar}. The method of distance computing is given in Algorithm 4, $T(v_i(s))$ denotes the tabu node set of the vertex v_i in solution s.

Algorithm 4 Distance_Compute(s_{ini}, s_{tar}).

```
1: Δ(s_ini, s_tar) ← 0
2: for (i = 1; i ≤ m; i++) do
3:    for (j = 1; j ≤ m; j++) do
4:       if T(v_i(s_tar)) == T(v_j(s_ini)) then
5:          if i == m then
6:             i ← 0
7:          end if
8:          if j == m then
9:             j ← 0
10:         end if
11:         if T(v_{i+1}(s_tar)) ≠ T(v_{j+1}(s_ini)) then
12:            Δ(s_ini, s_tar)++
13:         end if
14:      end if
15:   end for
16: end for
17: return Δ(s_ini, s_tar)
```

3.4.2. Trajectory exploration

To explore the trajectory from solution s_{ini} to solution s_{tar}, a series of neighborhood moves are performed on the initial solution s_{ini}, which introduces attributes of solution s_{tar} to initial solution progressively. During this procedure, a sequence of intermediate solutions are generated which make up the trajectory. Here an element-insertion move (i.e., *EI_move*) is used. It moves only one element which is required to be displaced at each time, and a series of such moves deterministically reduce the distance between two solutions.

An example of the EI relinking process is presented in Fig. 2, where each number represents a required node. To generate a path from the initial solution to the target one, we first compute the distance between them. Observe that the tabu node set pairs (3, 7), (8, 6), (6, 4) and (5, 1) in the initial solution are broken in the target, the distance is four. Then starting from the initial solution, it undergoes a series of EI moves that reduces its distance to the target solution. In Fig. 2, the number is in bold when its relative position is incorrect. For instance, moving node 4 after node 3 in the initial solution repairs the pair (3′, 4) and breaks (4, 5′) temporarily during the first EI moving action.

Algorithm 5 shows the EI path relinking procedure for the TCTSP. It aims to find the best solution s^* in the path between solution s_{ini} and s_{tar}, let $suc\_v_i(s)$ be the successor of a node in the tabu node set T_i, $e_i(s)$ denotes the edge between a node in tabu node set T_i and its follow-up node in solution s.

Algorithm 5 EI_Path_Relinking(s_{ini}, s_{tar}).

```
1: Distance_Compute(s_ini, s_tar)
2: f* ← min{f(s_ini), f(s_tar))}
3: s* ← argmin{f(s_ini), f(s_tar)}
4: while Δ(s_ini, s_tar) ≠ 0 do
5:    for (i = 1; i ≤ m − 1; i++) do
6:       Check whether the tabu sets of successors of the nodes
         belonging to the tabu set T_i are same
7:       if T(suc_v_i(s_ini)) ≠ T(suc_v_i(s_tar)) then
8:          if e_i(s_ini) and e_i(s_tar) are on the same cluster or inter-
            cluster edges then
9:             s′ ← EI_move(s_ini, s_tar)
10:            if f(s′) < f* then
11:               f* ← f(s′)
12:               s* ← s′
13:            end if
14:            s_ini ← s′
15:            Distance_Compute(s_ini, s_tar)
16:         end if
17:      end if
18:   end for
19: end while
20: return s*
```

3.4.3. Integrating ACO and EI path relinking

To use path relinking strategies within a metaheuristic, an elite set (*ES*) which contains the best and most diverse ACO solutions is needed. If the elite set has not yet reached its limited size, the intermediate solution is inserted to the pool. Otherwise, it enters the pool if it is better than the worst solution in elite set and can increase the elite diversity.

There are two widely used strategies to combine path relinking with metaheuristics (Resendel and Ribeiro, 2005). The one is to use path relinking for inter-optimization (*EI path relinking 1*), and the other is for post-optimization (*EI path relinking 2*). Their details are described as follows:

- EI path relinking 1 (*EIPR1*): EI Path relinking is periodically applied on an elite solution and a local optimum obtained after the local search.
- EI path relinking 2 (*EIPR2*): EI Path relinking is performed on all pairs of elite solutions, either after every ACO interation or after ACO terminates.

Algorithm 6 and 7 present the procedure of ACO incorporated with two EI path relinking strategies respectively.

	initial solution	target solution	distance
	7→8→**6**→**4**→5→**1**→2→3	1→2'→3'→4→5'→6→7'→8	4
	intermediate solution		
1st EI move	4→**7**→8→**6**→**5**→**1**→2→3	1→2'→3'→4→5'→6→7'→8	4
2nd EI move	4→5→**7**→8→**6**→**1**→2→3	1→2'→3'→4→5'→6→7'→8	3
3rd EI move	4→5→6→7→8→1→2→3	1→2'→3'→4→5'→6→7'→8	0

2 and 2' are in the same tabu node set, and the same situation in 3 and 3'、5 and 5'、7 and 7'.

Fig. 2. Relinking two solutions by element-insertion moves.

Algorithm 6 ACO_with_EIPR1().

```
Set parameters, initialize pheromone trails
while termination condition not met do
    s' ← Construct_Ant_Solutions( )
    s' ← Local_Search(s')
    if f(s') < f(s*) then
        s* ← s'
    end if
    Update_Pheromones( )
    Update the Elite set
end while
s'' ← EI_Path_Relinking(ES)
if f(s'') < f(s*) then
    s* ← s''
end if
return s*
```

Algorithm 7 ACO_with_EIPR2().

```
Set parameters, initialize pheromone trails
while termination condition not met do
    s' ← Construct_Ant_Solutions( )
    s' ← Local_Search(s')
    if iteration > |ES| then
        s'' ← Random_Select_Solution(ES)
        s' ← EI_Path_Relinking(s'', s')
    end if
    if f(s') < f(s*) then
        s* ← s'
    end if
    Update_Pheromones( )
    Update the Elite set
end while
return s*
```

4. Computational results

Assuming $\gamma = \max_{k\in\{1,\cdots m\}}(|T_k|)$ is the maximum size of the tabu node sets. Since the TCTSP is the same as the CTSP when $\gamma = 1$, we use two sets of instances to check the performance of the proposed algorithms:

- Set 1: six types of CTSP instances ranging from 101 to 1000 cities which were first generated by Mestria et al. (2013);
- Set 2: 10 instances ranging from 70 to 442 cities adopted from the CTSP instances in Mestria et al. (2013) and are converted to the TCTSP instances by applying a partitioning method to generate the tabu node sets.

The travel cost between every two cities is their rounding-off Euclidean distance. For each inter-cluster edge, the cost is increased by the penalization value which is set to the maximum cost among all costs of the edges multiplied by 10. Table 1 shows the information of two sets of instances. Columns 1–7 present the details of instance including the instance name, the identifier, the number of vertices (#*vertices*), the number of tabu node sets ($\#T_k$), the number of clusters ($\#V_k$), the maximum size of the tabu node set (γ), and the set index.

Table 1
The details of instances.

Instance name	ID	#*vertices*	$\#T_k$	$\#V_k$	γ	Set
10-lin318-1	I_1	318	318	10	1	1
10-pcb442-1	I_2	442	442	10	1	1
25-eil101-1	I_3	101	101	25	1	1
144-rat783-1	I_4	783	783	144	1	1
300-20-111-1	I_5	300	300	20	1	1
500-25-308-1	I_6	500	500	25	1	1
300-6-1	I_7	300	300	6	1	1
700-20-1	I_8	700	700	20	1	1
C1k.0-1	I_9	1000	1000	10	1	1
C1k.1-1	I_{10}	1000	1000	10	1	1
200-4-h-1	I_{11}	200	200	4	1	1
600-8-z-1	I_{12}	600	600	8	1	1
10-lin318-64-2	I_{13}	318	64	10	>1	2
9-st70-14-2	I_{14}	70	14	9	>1	2
10-kroB100-20-2	I_{15}	100	20	10	>1	2
10-pcb442-89-2	I_{16}	442	89	10	>1	2
10-pr439-88-2	I_{17}	439	88	10	>1	2
25-eil101-21-2	I_{18}	101	21	25	>1	2
25-gil262-53-2	I_{19}	262	53	25	>1	2
25-kroA100-20-2	I_{20}	100	20	25	>1	2
25-lin105-21-2	I_{21}	105	21	25	>1	2
25-rat99-20-2	I_{22}	99	20	25	>1	2

All metaheuristics we used were coded in C++ and tested on a PC with 3.1 GHz Intel Core i5 and 4GB RAM running on the Windows 7 operating system. For each instance, our algorithm was executed 10 times independently. Notice that the compared algorithms were tested on different computational platform, so we need to convert the running times of our algorithms to the running times obtained from the corresponding paper, the transformed time limit of our algorithms is 657.29 s, since Mestria et al. (2013) tested his heuristics on a PC with 2.83 GHz CPU and the time limit was set as 720 s in Mestria et al. (2013).

4.1. Parameter setting

There are four parameters in ACO: the number of ants N, the relative importance of pheromone and heuristic factors a, b, and the pheromone evaporation ratio ρ. The parameter values are determined using statistical study. The candidate values of these parameters are $N = \{10, 20, 30, 50, 100\}$, $a = \{0.5, 1, 2, 5, 10\}$, $b = \{0.05, 0.1, 0.5, 1, 3\}$, $\rho = \{0.3, 0.5, 0.7, 0.9, 0.95\}$. To analyze the effect of each parameter, only one parameter is varied while the oth-

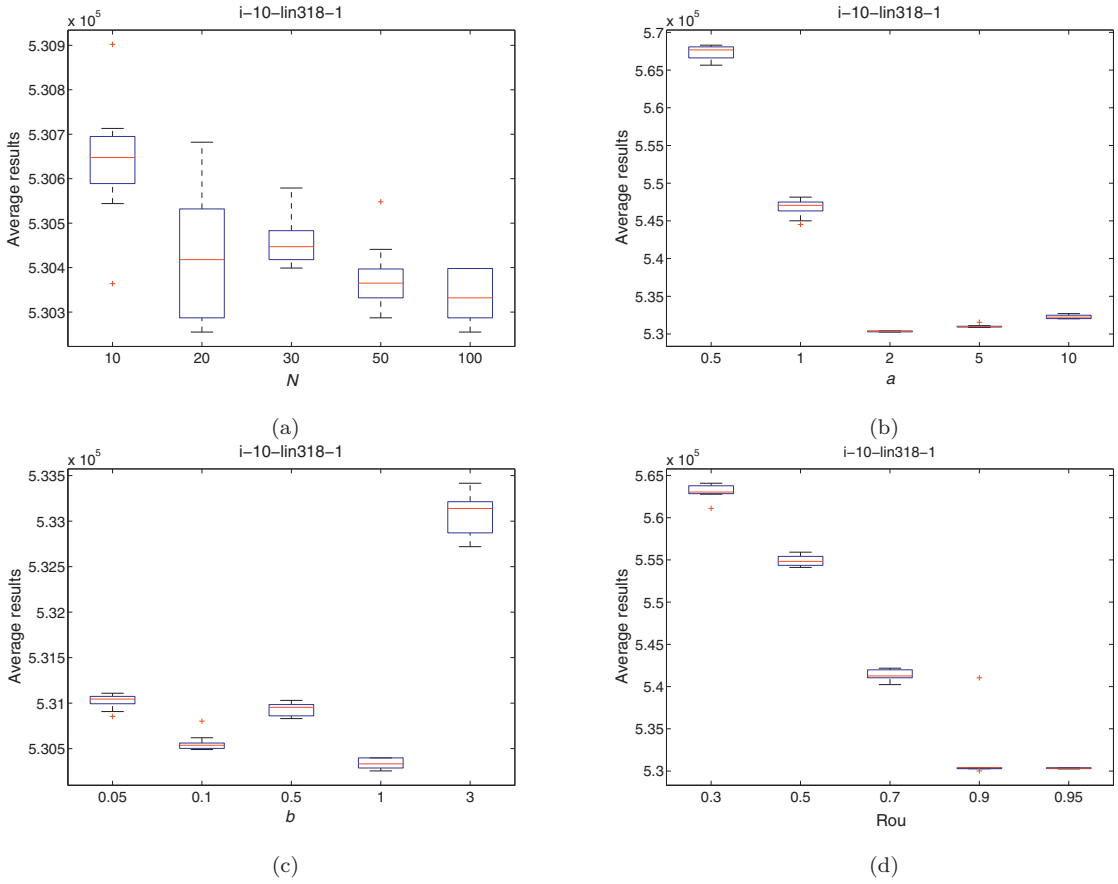

Fig. 3. Parameter test. (a)N, (b)a, (c)b, (d)ρ (i.e., Rou).

ers are held constant. Fig. 3 presents the average results obtained on instance I_1, one can notice that ACO works well by choosing the default following values: $N = 100, a = 2, b = 1, \rho = 0.95$.

The parameters used for GRASP and Path Relinking were taken from Mestria et al. (2013). The value of restricted candidate list control parameter α used for GRASP is set using a reactive strategy, the candidate value of α are {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0} and the probability distribution of α is updated every 25 iterations. The size of the elite set used for path relinking is set to 10, the minimum difference between a new candidate solution and any solution in the elite set is set to 5.

4.2. Results obtained by the proposed metaheuristics

We first compare the results obtained on all instances of Set 2:

- *ACO* : The proposed Ant colony Optimization method with 2-Opt local search;
- *AEIPR1* : *ACO* with element-insertion Path Relinking EIPR1;
- *AEIPR2: ACO* with element-insertion Path Relinking EIPR2;
- *AEIPR1R2* : *ACO* with EIPR1 and EIPR2;
- *GRASP* : Traditional GRASP heuristic (Mestria et al., 2013);
- *GEIPR1* : *GRASP* with element-insertion Path Relinking EIPR1;
- *GEIPR2* : *GRASP* with element-insertion Path Relinking EIPR2;
- *GEIPR1R2* : *GRASP* with EIPR1 and EIPR2.

Table 2 and 3 report the best and average results of these algorithms from columns 2–9. The best values are shown in boldface. The last row presents the average values of the results. As seen from Table 2, AEIPR1R2 obtains the best results for five instances, AEIPR1 gets one. So in terms of the numbers of best solutions, AEIPR1R2 is the best.

According to Table 3, nine best average results are obtained by AEIPR1R2, one by AEIPR1. One can observe that AEIPR1R2 obtains the best average and the values got by the other ACO heuristics are smaller than the values obtained by GRASP heuristics.

4.3. Comparison with the exsiting algorithms

Owing to the competitive performance of ACO, in this Section, we compare the proposed ACO with three GRASP algorithms, denoted as G, GPR1, GPR2, GPR1R2 in Mestria et al. (2013) for instances of Set 1. Moreover, since the adaptive large neighborhood search(ALNS) has shown to be the current state-of-the-art for different routing problems (Pisinger and Ropke, 2005; Vidal et al., 2012, 2013), we also have extended it to deal with the TCTSP. The details of the extended ALNS are presented in Appendix. These algorithms are compared according to gap_{lb} which is calculated as $100 \times \frac{value-lb}{value+\epsilon}$, where *value* is the result found by the corresponding algorithm, *lb* the lower bound and ϵ is equal to 10^{-10} (Mestria et al., 2013).

Table 4 reports the gap_{lb} values for the best results obtained in 10 executions. The first column shows the instance identifiers, and the following columns report the values for gap_{lb} found by the ACO, GRASP heuristics and the ALNS. The bold values in Table 4 indicate the best values. As seen from Table 4, the proposed ACO heuristics outperform or are equal to the other approaches on eleven problem instances, and ALNS obtains the best results for one. The best average gap_{lb} values of GRASP heuristics is 0.82%, while the same values for AEIPR1R2 and ALNS are 0.57% and 0.63%, respectively.

Table 5 presents the gap_{lb} values for the average results obtained. The first column shows the instance identifiers and the last

Table 2
Best results obtained on the instances of Set 2 with limited processing time.

ID	Best solution values							
	ACO	AEIPR1	AEIPR2	AEIPR1R2	GRASP	GEIPR1	GEIPR2	GEIPR1R2
I_{13}	509412.00	509288.00	509248.00	**508523.00**	510021.00	509544.00	509532.00	509411.00
I_{14}	**9346.00**	**9346.00**	**9346.00**	**9346.00**	**9346.00**	**9346.00**	**9346.00**	**9346.00**
I_{15}	427148.00	**427083.00**	427133.00	427113.00	427185.00	427185.00	427171.00	427171.00
I_{16}	510449.00	510214.00	510200.00	**509411.00**	510566.00	510467.00	510436.00	510237.00
I_{17}	1348138.00	1347103.00	1346841.00	**1346312.00**	1348319.00	1347783.00	1347499.00	1347256.00
I_{18}	**10411.00**	**10411.00**	**10411.00**	1**0411.00**	10412.00	**10411.00**	**10411.00**	1**0411.00**
I_{19}	54365.00	54313.00	54341.00	**54290.00**	54389.00	54374.00	54381.00	54364.00
I_{20}	508302.00	508257.00	508248.00	**508245.00**	508611.00	508609.00	508435.00	508435.00
I_{21}	518497.00	518497.00	**518487.00**	**518487.00**	518532.00	518527.00	518527.00	518522.00
I_{22}	**26674.00**	**26674.00**	**26674.00**	**26674.00**	26678.00	26677.00	26677.00	26675.00
Average value	392274.20	392118.60	392092.90	**391881.20**	392405.90	392292.30	392241.50	392182.80

Table 3
Average results obtained on the instances of Set 2 with limited processing time.

ID	Average solution values							
	ACO	AEIPR1	AEIPR2	AEIPR1R2	GRASP	GEIPR1	GEIPR2	GEIPR1R2
I_{13}	509548.20	509570.30	509531.60	**508671.10**	510236.40	509979.90	509857.10	509805.10
I_{14}	9346.90	9346.50	9346.30	**9346.00**	9347.60	9347.40	9347.30	9347.20
I_{15}	427163.60	**427105.90**	427146.50	427128.60	427246.90	427237.60	427233.40	427225.00
I_{16}	511074.20	510983.60	510317.90	**509516.00**	511029.50	510950.80	510782.40	510621.30
I_{17}	1348954.90	1348106.20	347350.70	**1347246.80**	1349095.80	1349150.10	1348182.10	1348016.90
I_{18}	10411.20	**10411.00**	**10411.00**	**10411.00**	10412.60	10412.10	10412.80	10412.20
I_{19}	54375.70	54327.20	54347.10	**54300.00**	54402.10	54392.10	54396.90	54393.40
I_{20}	508331.40	508323.70	508276.20	**508260.90**	508768.90	508736.80	508734.80	508655.10
I_{21}	518507.00	518504.70	518501.00	**518492.80**	518564.90	518561.40	518557.60	518542.10
I_{22}	26675.10	26674.90	26674.60	**26674.10**	26680.50	26679.20	26679.70	26678.90
Average value	392438.82	392335.40	392190.29	**392004.73**	392578.52	392544.74	392418.41	392369.72

Table 4
Best results obtained on the instances of Set 1 with limited processing time.

ID	Best solution values								
	ACO	AEIPR1	AEIPR2	AEIPR1R2	G	GPR1	GPR2	GPR1R2	ALNS
I_1	0.77	**0.72**	0.75	0.73	0.97	0.80	0.93	0.76	0.84
I_2	0.60	0.48	0.43	0.46	1.12	0.73	0.99	0.66	**0.37**
I_3	0.02	**0.01**	**0.01**	**0.01**	0.04	0.04	0.05	0.03	**0.01**
I_4	0.12	0.09	0.10	**0.08**	0.21	0.19	0.21	0.20	**0.08**
I_5	0.50	0.44	0.47	**0.36**	0.55	0.45	0.51	0.43	0.44
I_6	0.53	**0.51**	**0.51**	**0.51**	0.69	0.61	0.66	0.58	0.53
I_7	0.34	0.29	**0.26**	0.28	0.67	0.51	0.60	0.49	0.43
I_8	0.52	0.51	0.49	**0.46**	0.67	0.63	0.65	0.64	0.58
I_9	1.40	1.34	1.36	**1.13**	1.63	1.61	1.64	1.60	1.24
I_{10}	1.11	0.96	0.96	**0.88**	1.28	1.27	1.31	1.27	0.93
I_{11}	0.85	0.77	**0.66**	0.78	1.68	1.28	1.60	1.19	0.84
I_{12}	1.34	1.22	1.18	**1.13**	2.23	1.80	2.14	1.96	1.24
Average value	0.67	0.61	0.60	**0.57**	0.98	0.83	0.94	0.82	0.63

Table 5
Average results obtained on the instances of Set 1 with limited processing time.

ID	Average solution values								
	ACO	AEIPR1	AEIPR2	AEIPR1R2	G	GPR1	GPR2	GPR1R2	ALNS
I_1	0.78	0.74	0.76	**0.73**	1.78	1.18	1.21	1.18	0.87
I_2	0.62	0.53	**0.45**	0.50	1.93	1.24	1.36	1.22	0.48
I_3	0.03	0.02	**0.01**	**0.01**	0.35	0.22	0.14	0.18	0.02
I_4	0.13	0.10	0.11	**0.09**	0.28	0.23	0.25	0.24	0.10
I_5	0.51	0.45	0.47	**0.36**	0.85	0.64	0.63	0.63	0.48
I_6	0.54	0.53	0.54	**0.52**	0.92	0.73	0.78	0.73	0.56
I_7	0.36	0.30	0.28	**0.27**	1.09	0.82	0.78	0.78	0.48
I_8	0.54	0.53	0.51	**0.48**	0.82	0.71	0.75	0.72	0.64
I_9	1.46	1.42	1.44	**1.21**	1.88	1.76	1.79	1.76	1.41
I_{10}	1.15	**1.01**	1.02	1.03	1.54	1.44	1.47	1.42	1.13
I_{11}	0.92	0.79	**0.69**	0.82	3.30	2.37	2.01	2.30	0.88
I_{12}	1.40	1.31	1.29	**1.23**	3.32	2.43	2.70	2.54	1.45
Average value	0.70	0.64	0.63	**0.61**	1.51	1.15	1.16	1.14	0.71

nine columns show the gap_{lb} values. The bold values indicate the best values. The last line presents the average of gap_{lb} values. We can observe that ACO heuristics found better average results than both GRASP heuristics and the ALNS. AEIPR1R2 obtains most of the best average results. The best average gap_{lb} values of AEIPR1R2 and ALNS are 0.61% and 0.71%, while the best average values of GPR1R2 is equal to 1.14%.

The results shown in Tables 4 and 5 demonstrate that the proposed ACO heuristics are able to find better solutions than both GRASP heuristics and the ALNS under the same time limit, so ACO heuristics may be more suitable for solving these instances.

5. A real-life application of the TCTSP

TT&C resources scheduling plays a vital role in the management of satellite network. This paper considers the ground-space integrated resources scheduling of Medium Earth Orbit (MEO) satellites in Compass GNSS of China. Due to these practical characteristics, this problem is called MEO-ITTCRSP. The TCTSP is a minimization problem, whereas the MEO-ITTCRSP aims to maximize the total scheduling profit, we may converse the MEO-ITTCRSP into a minimization one by minimizing the negative values of the objective. As there are some differences between the MEO-ITTCRSP and the TCTSP, in order to apply the ACO heuristics proposed for the TCTSP, we first transform the MEO-ITTCRSP to the TCTSP.

5.1. Transformation from the MEO-ITTCRSP to the TCTSP

The MEO-ITTCRSP involves several missions, each mission belongs to an exclusive satellite and is composed of a set $J(i) = \{j_1, \cdots, j_{|J(i)|}\}$ of satellite services. A satellite service consists of a set $S = \{s_1, \cdots, s_{|S|}\}$ of satellites and a set $E = \{e_1, \cdots, e_{|E|}\}$ of ground stations. In the integrated TT&C resources scheduling problem, a ground station e_i can communicate with a satellite s_j directly or using a relay satellite $relay\_s_j$ which lies in its field of view to retransmit messages. Therefore, for each $(s_j, e_i) \in S \times E$, the visibility period can be expressed as the set $A(s_j, e_i) = \{a_1, \cdots, a_{|A(s_j,e_i)|}\}$ of disjoint contact opportunity windows which are made up of direct contact opportunity windows (i.e., ground-based resources) and indirect contact opportunity windows (i.e., space-based resources). Each contact opportunity window a_i is characterized by the following parameters:

- aS_i, the (unique) satellite establishing communication in a_i;
- aE_i, the (unique) ground station establishing communication in a_i;
- aJ_i, the service which can be processed with a_i;
- $[aW_i^s, aW_i^e]$, the time window of a_i;
- $relay\_s_i$, the relay satellite (i.e., a satellite that has a radio relay station on board, the radio relay station can receive messages and retransmit or rebroadcast them). If it is the same as aS_i, a_i is a direct contact opportunity window (i.e., aE_i applies service to s_i directly), otherwise an indirect contact opportunity window (i.e., indirectly service via $relay\_s_i$ is executed).
- aPr_i, the profit of a_i.

In the MEO-ITTCRSP, a set $J = \{J(1), \cdots, J(|S|)\}$ of services has to be scheduled for the requirement of satellite control operations. Each service J_j specifies a minimum communication duration $tlast_j$ and can only be processed in a given time window $TW_j = [TW_j^s, TW_j^e]$, where TW_j^s, TW_j^e are lower and upper bounds of the time window, respectively. The service can not be preempted once it is assigned with a contact opportunity window a_i, the corresponding profit aPr_i will be earned when the service finished. Moreover, the interval time of any two successive services of each satellite must be between the minimum and the maximum interval time, denoted as Δt_{min} and Δt_{max}, respectively. The problem aims to find a feasible schedule Ω in which the contact opportunity windows are assigned to as many services as possible such that the total received profit F is maximized.

In many practical applications, control center must serve a satellite several times in a scheduling period in order to determine its orbit. For example, to meet the accuracy requirement of orbit determination, a MEO satellite must be served 4 times which are evenly distributed throughout a day. For this reason, a scheduling period is always partitioned into four equal time slots and the visible arcs are often divided into four clusters $A_i (A_i \subset A, i = 1, \cdots, 4, \bigcup_i A_i = A, A_i \bigcap A_j = \emptyset, \forall A_i, A_j \subset A)$ by the time slot it belongs to. Moreover, according to the resource constraint, namely, each ground station can serve only one satellite and each satellite requires only one service at a time, so any two time-overlapping contact opportunity windows which belong to the same ground station or serve the same satellite conflict with each other and are put in the same conflict contact opportunity window set $C_i (C_i \subset A, i = 1, \cdots, p, \bigcup_i C_i = A)$. When a pair of contact opportunity windows conflict with each other, a penalization value is added to the edges connecting them. Such that infeasible solutions become unattractive relative to the optimum solution and the algorithm treats these poor solutions as non-optimal ones.

Based on the above description, we observe that the MEO-ITTCRSP with these specialized constraints can be solved as a TCTSP. However, we should do some transformations first. Table 6 gives a summary of the one-to-one correspondence between the MEO-ITTCRSP and the TCTSP. The symbols used for the TCTSP are denoted in Section 2.

5.2. The definition of the model

For the sake of clarity, this section presents an integer programming formulations for the MEO-ITTCRSP. We introduce a binary variable x_i assuming value 1 if contact opportunity window a_i is assigned to service aJ_i, otherwise 0. Hence, The MEO-TTCRSP can be stated as follows:

$$\max \quad F(X) = \sum_{|A|} x_i \cdot aPr_i \tag{12}$$

$$\text{s.t.} \quad TW_j^s \le aW_i^s \le aW_i^e \le TW_j^e, \forall J_j \in J, \exists a_i \in A, aJ_i = J_j, x_i = 1 \tag{13}$$

$$\sum_{a_i \in C_k} x_i = 1, \forall k \in \{1, \cdots, p\} \tag{14}$$

$$\sum_{|A|} x_i \ge 4, \forall a_i \in A, \forall s_j \in S, aS_i = s_j \tag{15}$$

$$\Delta t_{min} \le aW_n^s - aW_m^e \le \Delta t_{max}, \exists a_m \in A, x_m = 1, \exists a_n \in A, \tag{16}$$

$$aW_n^s = min\{aW_i^s | \exists a_i \in A, x_i = 1, aS_i = aS_m, aW_i^s - aW_m^e > 0\}$$
$$x_i \in \{0, 1\}, \forall a_i \in A \tag{17}$$

Formulation (12) states the objective function. Constraints (13) ensure that the service must be executed in the valid time window. Constraints (14) ensure that only one of a conflict contact opportunity window set can be chosen to execute its corresponding service. Constraints (15) and (16) are the basic requirements of service constraints, (15) means the number of service times offered must be not less than the minimum number of required service times, (16) states the maximum and minimum interval time constraints between any two successive services of the same satellite. Constraints (17) impose the restriction on the decision variables.

Table 6
Summary of the one-to-one correspondence between the MEO-ITTCRSP and TCTSP.

TCTSP		MEO-ITTCRSP	
Element	Symbol	Element	Symbol
Node	v_i	Visible arc	a_i
Edge	(v_i, v_j)	Conflict free arc pair	(a_i, a_j)
Cluster	V_i	visible arcs in ith interval	A_i
Tabu node set	T_i	Conflict visible arc set	C_i
Edge cost	c_{ij}	Visible arc profit	aPr_i
Path	S	Scheduling plan	X
Total distance traveled by salesman	z	Total earned profits	F

Table 7
The basic information of the practical instances.

Problem instance name	Scheduling period	#services	#satellites	#relay satellites	#ground stations	# contact opportunity window
CMEO1	2012/7/1-2	96	24	0	4	1773
CMEO2	2012/7/1-2	96	24	24	4	8865

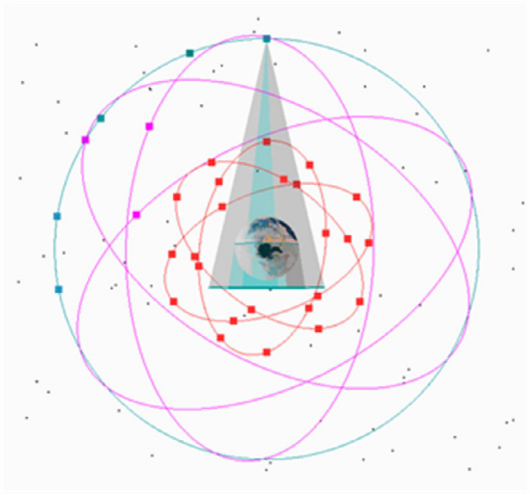

Fig. 4. The layout of Compass GNSS.

5.3. Case studies

5.3.1. The instances

After the regional navigation system of China began service, Compass GNSS develops rapidly. As a result, the number of satellites in orbit is growing with great speed and the disadvantages of ground-based TT&C network such as low coverage, high maintenance costs and disability of multi-satellite TT&C services become impossible to ignore. To make full use of scarce satellite resources, looking for an effective and practical scheduling algorithm becomes necessary and urgent.

Compass GNSS consists of 30 satellites: 24 MEO satellites which form a *Walker* 24/3/1 constellation, 3 Geosynchronous Earth Orbit (GEO) satellites, and 3 Inclined Geosynchronous Satellite Orbit (IGSO) satellites, the layout of Compass GNSS is shown in Fig. 4, where red, violet, and blue squares represent the MEO satellites, IGSO satellites and GEO satellites, respectively.

5.3.2. Computational results

We consider two instances designed for the orbit determination services scheduling of the MEO satellites in Compass GNSS. The basic information is given in Table 7. Without loss of generality, the long contact opportunity window of MEO satellite is divided into short time windows by the standard of the minimum duration of all services in order to make good use of these contact opportunity windows. Moreover, using 4 segment orbit determination method described in Section 5.1, all short contact opportunity windows are divided into 4 clusters corresponding 4 time slots. In summary, within each time slot, each satellite needs to be served with no less than 30 min duration. The profit of contact opportunity windows are shown in Table 8.

Table 8
The profit of visible arcs.

Visible arc type	Scheduling profit
Ground-satellite contact opportunity window	1.00
Inter-orbit plane inter-satellite contact opportunity window	0.95
Intra-orbit plane inter-satellite contact opportunity window	0.90

Table 9
Comparison among different scheduling algorithms.

Instance	AEIPR1R2		GEIPR1R2		GA	
	Best	Average	Best	Average	Best	Average
CMEO1	**76.00**	**75.50**	69.00	68.80	72.00	70.10
CMEO2	**94.20**	**93.87**	91.95	89.02	92.25	91.91

To study the performance of AEIPR1R2, experiment is executed to compare it with GEIPR1R2 and genetic algorithm (GA) which was applied to the same real-life instances in Tianjiao et al. (2014). The stopping criterion for the algorithms is still a time limit of 657.29 s. Based on our preliminary test, the parameters for AEIPR1R2 are set to $N = 100, a = 1, b = 1/3, \rho = 0.85$. For GEIPR1PR2, the parameters are set the same as in Section 4.1. The number of executions is set to 10 for each instances for all algorithms.

The best and average solution results are given in Table 9. The first column shows the instance identifiers, followed by the results obtained by the algorithms. The bold values indicate the best values.

According to Table 9, AEIPR1R2 is better than other two heuristics in terms of both the best and the average values.

A scheduling plan of CMEO2 with the best result found by AEIPR1R2 is displayed with Gantt chart in Fig. 5, where filled rectangles represent that the TT&C services are offered by ground stations directly and the hollow rectangles are the indirect services applied via relay satellite. The number above the rectangle is the identifier of the ground station which provides the corresponding service.

Fig. 5 shows that every satellite is served 4 times which are evenly distributed in four time slots throughout the scheduling period, and the burden of each ground station is balanced.

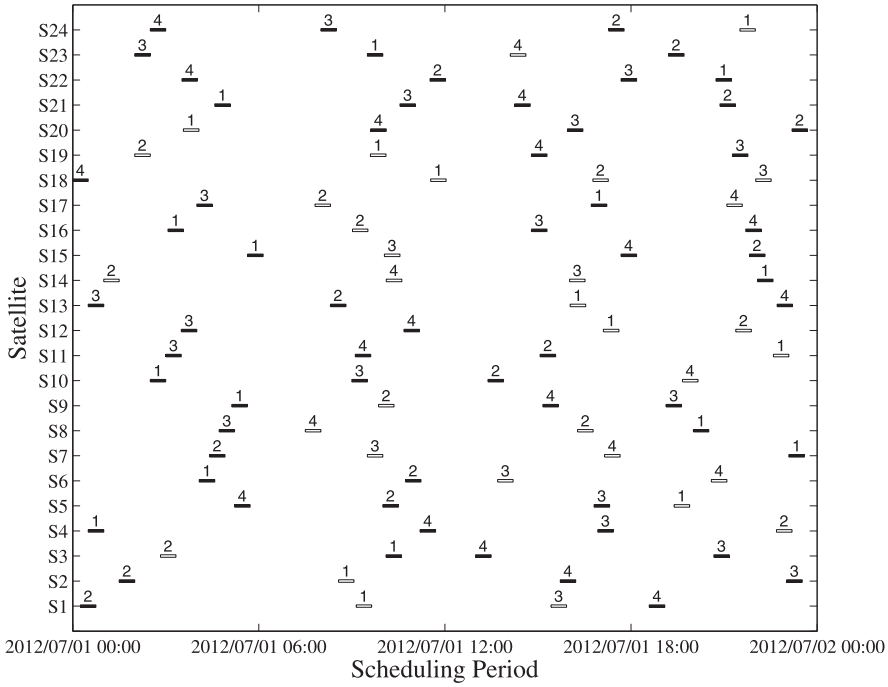

Fig. 5. The Gantt chart of the best scheduling plan of CMEO2.

6. Conclusions

Motivated by a real-life application in China Satellite Control Center, this paper considers a new variant of the CTSP, called tabu clustered traveling salesman problem (TCTSP). We extend both ACO and GRASP heuristics from TSP to this field. Several improved path relinking strategies incorporated to these two kinds of metaheuristics are investigated. The computational results obtained by eight proposed heuristics show that all algorithms are able to get solutions with good quality for the TCTSP in a reasonable time and that the heuristic AEIPR1R2 outperforms the other heuristics proposed in this paper. Moreover, a fair comparison is made among our ACO heuristics, three GRASP heuristics and ALNS developed for solving the TCTSP. According to computational results, our heuristics achieve very promising results. Especially, it can find new best solutions for all chosen CTSP benchmark instances.

We also formulate a class of specialized TT&C Resources Scheduling Problem and show the transformation from the specialized TTCRSP to the TCTSP. The proposed approach offers a possibility to handle other variations of the TTCRSP as well as some other potential constraints, which is a significant requirement of aerospace systems.

Acknowledgments

Thanks are due to the State Key Laboratory of Astronautic Dynamics for their supporting this work and Xi'an Satellite Control Center for providing practical TT&C cases. This work is supported by the Key Project of National Natural Science Foundation of China (NO. 91638202), the National Natural Science Foundation of China (NO. 11503096), the National Natural Science Foundation of China (NO. 61573277), the Open Projects Program of National Laboratory of Pattern Recognition, the National Key Basic Research and Development Program (NO. 613237).

Appendix A

The ALNS explores the search space by using a number of competing sub-heuristics, each of which is associated with a frequency corresponding to its historic performance. The process of the ALNS can be viewed as a sequence of remove-insert operations. The extended ALNS for the TCTSP follows the basic framework of ALNS in Ropke and Pisinger (2006a), and it introduces some changes in removal and insertion heuristics so as to make them suitable to the problem at hand. The extended ALNS is a two phase metaheuristic. In the first phase, an initial solution is constructed using the GRASP proposed in this work. In the second phase, the obtained solution is gradually improved by alternately destroying and repairing the solution. Specifically, a number of solution elements are removed from the current solution by a chosen removal heuristic, then an insertion heuristic is selected to reinsert the elements into the current solution again. Multiple removal and insertion heuristics are available to be selected. The selection of removal/insertion methods is controlled dynamically according to their recorded performance.

Algorithm 8 shows the extended ALNS procedure, adapted from the ALNS flowchart in Pisinger and Ropke (2010). Three variables are maintained by the algorithm. The variable s^* is the best solution found during the search, s is the current solution and s' is a temporary solution that may be discarded or improved to the status of current solution. The function $d(\cdot)$ is the destroy method while $r(\cdot)$ is the repair one. The sets of destroy and repair methods are denoted by Ω^- and Ω^+, respectively. Two vectors ρ^- and ρ^+ are used to store the weight of each destroy and repair method. In line 3, the weight vectors ρ^- and ρ^+ are used to select the destroy and repair methods using a roulette-wheel principle. In line 5, a simulated annealing based acceptance criteria is employed. When an iteration of the ALNS heuristic is completed, the weight vectors are updated according to the adaptive weight adjustment rule (Ropke and Pisinger, 2006a). More details about the ALNS can be found in Pisinger and Ropke (2005), Ropke and Pisinger (2006a), Pisinger and Ropke (2010) and Ropke and Pisinger (2006b).

Algorithm 8 Adaptive large neighborhood search ($s \in \{solution\}, \rho^- = (1, \ldots, 1), \rho^+ = (1, \ldots, 1)$).

$s^* \leftarrow s$
repeat
  Select destroy $d \in \Omega^-$ and repair methods $r \in \Omega^+$ using ρ^- and ρ^+
  $s' = r(d(s))$
  if $accept(s', s)$ **then**
    $s \leftarrow s'$
  end if
  if $f(s) < f(s^*)$ **then**
    $s^* \leftarrow s$
  end if
  Update ρ^- and ρ^+
until stop criterion is met
return s^*

Three different removal heuristics are used in our extended ALNS. Each removal heuristic has its own strategy for choosing q elements to remove. The heuristics are:

- Random removal heuristic: The elements are chosen at random.
- Worst removal heuristic: Remove the elements so as to decrease the cost the most.
- Cluster removal heuristic: Remove the clustered elements.

We adopted the first two heuristics directly from Ropke and Pisinger (2006a), and the last one is modified from the one in Ropke and Pisinger (2006a). The cluster removal heuristic tries to remove clustered cities from the route in the TCTSP since the cities in a same cluster are related in some sense and hence easy to interchange. Given a city i in a solution s, we define the cost of the city as $cost(i, s) = f(s) - f_{-i}(s)$, where $f_{-i}(s)$ is the objective value of the solution without city i. The algorithm initially selects a cluster randomly, then the cities in the chosen cluster are removed. If less than q cities have been removed, we pick and remove cities from another cluster, which are most related to the randomly chosen cities in the removed cluster. The relatedness measure used in this paper only depends on the distance between two cities. The smaller the distance c_{ij} is, the more related the two cities are. The process continues until the desired number of cities has been removed from the route. The cluster removal heuristic is shown in pseudo-code in Algorithm 9.

Algorithm 9 Cluster removal ($s \in \{solution\}, q \in N, p \in R_+$).

Randomly select a cluster of cities C_i in s
Set of cities: $D = C_i$
while $|D| > q$ **do**
  Array: $L =$ all cities $i \in D$
  Sort L by descending $cost(i, s)$
  Choose a random number y in the interval $[0, 1)$
  City: $r = D[y^p|D|]$
  $D = D \backslash \{r\}$
end while
while $|D| < q$ **do**
  City: $r =$ a randomly selected city from D
  Array: $L =$ an array containing all cities from s not in D
  Sort L such that $i < j \Rightarrow c_{r,L[i]} < c_{r,L[j]}$
  Choose a random number y in the interval $[0, 1)$
  $D = D \cup \{L[y^p|L|]\}$
end while
remove the cities in D from s

In order to insert cities to the route, we followed the idea of insertion heuristics proposed in Ropke and Pisinger (2006a). Note that there is only one route in each feasible solution of our problem. We use the insertion cost $insertionC_i^j$ to denote the change in objective value incurred by inserting city i into the position which has the $j-th$ lowest insertion cost. If the insertion violates the constraints of the TCTSP, then we set $insertionC_i = \infty$. In $regret-k$ heuristic, we define the regret value $r_i^k = insertionC_i^k - insertionC_i^1$. Insertion heuristics for the TCTSP are described as follows:

- Basic greedy insertion heuristic: In each iteration one removed city is inserted into the route such that the cost will be increased the least.
- Regret-2 insertion heuristic: It chooses the city i with the largest regret value r_i^2 to insert at the position such that the least increment of the cost will be incurred.
- Regret-3 insertion heuristic: It is similar to the regret-2 heuristic except selecting the city i with the maximal regret value r_i^3.

We implemented the ALNS algorithm in the C++ programming language and executed on the same computer described in Section 4. The parameter values used in ALNS are chosen according to statistical results. The complete parameter tuning resulted in the following parameter vector $(p, w, c, \sigma_1, \sigma_2, \sigma_3, r) = (3, 0.05, 0.9998, 33, 9, 13, 0.1)$, where σ_1, σ_2, σ_3, r are used in the weight adjustment algorithm, and the acceptance criteria is controlled by two parameters: w, c. The start temperature is set such that a solution that is $w+1$ times larger than z is accepted with probability 0.5 when the objective value of the current solution is z. The number of cities to remove is found as a random number between $min\{0.1n, 30\}$ and $min\{0.4n, 60\}$. That is, the number will be in the interval $[0.1n, 0.4n]$ for small instances, while the interval is [30, 60] for larger ones.

References

Barbulescu, L., Watson, J.-P., Whitley, L.D., Howe, A.E., 2004. Scheduling space—ground communications for the air force satellite control network. J. Schedul. 7 (1), 7–34.
Bianchessi, N., Cordeau, J.-F., Desrosiers, J., Laporte, G., Raymond, V., 2007. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. Eur. J. Oper. Res. 177 (2), 750–762.
Chisman, J.A., 1975. The clustered traveling salesman problem. Comput. Oper. Res. 2 (2), 115–119.
Dorigo, M., Birattari, M., 2010. Ant colony optimization. In: Encyclopedia of Machine Learning. Springer, pp. 36–39.
Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. Syst. Man Cybern. Part BIEEE Trans. 26 (1), 29–41.
Feo, T.A., Resende, M.G., Smith, S.H., 1994. A greedy randomized adaptive search procedure for maximum independent set. Oper. Res. 42 (5), 860–878.
Gendreau, M., Laporte, G., Potvin, J.-Y., 1994. Heuristics for the Clustered Traveling Salesman Problem. Technical Report.
Glover, F., 1997. Tabu search and adaptive memory programming advances, applications and challenges. In: Interfaces in Computer Science and Operations Research. Springer, pp. 1–75.
Gutin, G., Punnen, A.P., 2002. The traveling salesman problem and its variations, vol. 12. Springer Science & Business Media.
Jongens, K., Volgenant, T., 1985. The symmetric clustered traveling salesman problem. Eur. J. Oper. Res. 19 (1), 68–75.
Karapetyan, D., Minic, S.M., Malladi, K.T., Punnen, A.P., 2015. Satellite downlink scheduling problem: a case study. Omega 53, 115–123.
Laporte, G., Palekar, U., et al., 2002. Some applications of the clustered travelling salesman problem. J. Oper. Res. Soc. 53 (9), 972–976.
Laporte, G., Potvin, J.-Y., Quilleret, F., 1997. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. J. Heuristics 2 (3), 187–200.
Lokin, F., 1979. Procedures for travelling salesman problems with additional constraints. Eur. J. Oper. Res. 3 (2), 135–141.
Marinelli, F., Nocella, S., Rossi, F., Smriglio, S., 2011. A lagrangian heuristic for satellite range scheduling with resource constraints. Comput. Oper. Res. 38 (11), 1572–1583.
Mestria, M., Ochi, L.S., de Lima Martins, S., 2013. Grasp with path relinking for the symmetric euclidean clustered traveling salesman problem. Comput. Oper. Res. 40 (12), 3218–3229.
Pisinger, D., Ropke, S., 2005. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.
Pisinger, D., Ropke, S., 2010. Large Neighborhood Search.
Potvin, J.-Y., Guertin, F., 1996. The clustered traveling salesman problem: a genetic approach. In: Meta-Heuristics. Springer, pp. 619–631.

Potvin, J.-Y., Guertin, F., 1998. A genetic algorithm for the clustered traveling salesman problem with a prespecified order on the clusters. In: Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search. Springer, pp. 287–299.

Reghioui, M., Prins, C., Labadi, N., 2007. Grasp with path relinking for the capacitated arc routing problem with time windows. In: Applications of Evolutionary Computing. Springer, pp. 722–731.

Resendel, M.G., Ribeiro, C.C., 2005. Grasp with path-relinking: recent advances and applications. In: Metaheuristics: Progress as Real Problem Solvers. Springer, pp. 29–63.

Ropke, S., Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Ropke, S., Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. Oper. Res. 171 (3), 750–775.

Stützle, T., Hoos, H.H., 2000. Max–min ant system. Future Gen. Comput. Syst. 16 (8), 889–914.

Tianjiao, Z., Zexi, L., Jing, L., 2014. An adaptive genetic algorithm for solving ground-space tt&c resources integrated scheduling problem of beidou constellation. In: Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese. IEEE, pp. 1785–1792.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60 (3), 611–624.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur. J. Oper. Res. 231 (1), 1–21.

Wu, G., Liu, J., Ma, M., Qiu, D., 2013. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Comput. Oper. Res. 40 (7), 1884–1894.

Zhang, Z., Zhang, N., Feng, Z., 2014. Multi-satellite control resource scheduling based on ant colony optimization. Expert Syst. Appl. 41 (6), 2816–2823.