Pintea *et al. Complex Adapt Syst Model (2017) 5:8*
DOI 10.1186/s40294-017-0048-9

Complex Adaptive Systems Modeling

RESEARCH Open Access

The generalized traveling salesman problem solved with ant algorithms

Camelia-M. Pintea*†, Petrică C. Pop† and Camelia Chira†

*Correspondence:
dr.camelia.pintea@ieee.org
†Camelia-M. Pintea, Petrică C. Pop and Camelia Chira contributed equally to this manuscript
Technical University of Cluj-Napoca, North University Center, Baia-Mare, Romania

Abstract

A well known $\mathcal{NP}$-hard problem called the *generalized traveling salesman problem* (*GTSP*) is considered. In *GTSP* the nodes of a complete undirected graph are partitioned into clusters. The objective is to find a minimum cost tour passing through exactly one node from each cluster. An exact exponential time algorithm and an effective meta-heuristic algorithm for the problem are presented. The meta-heuristic proposed is a modified *Ant Colony System* (*ACS*) algorithm called *reinforcing Ant Colony System* which introduces new correction rules in the *ACS* algorithm. Computational results are reported for many standard test problems. The proposed algorithm is competitive with the other already proposed heuristics for the *GTSP* in both solution quality and computational time.

Keywords: Meta-heuristics, Intelligent agents, Graphs, Data clustering

Background

Many combinatorial optimization problems are $\mathcal{NP}$-hard, and the theory of $\mathcal{NP}$-completeness has reduced hopes that *$\mathcal{NP}$-hard* problems can be solved within polynomially bounded computation times (Dahlke 2008; Dunne 2008). Nevertheless, sub-optimal solutions are sometimes easy to find. Consequently, there is much interest in approximation and heuristic algorithms that can find near optimal solutions within reasonable running time. Heuristic algorithms are typically among the best strategies in terms of efficiency and solution quality for problems of realistic size and complexity.

In contrast to individual heuristic algorithms that are designed to solve a specific problem, meta-heuristics are strategic problem solving frameworks that can be adapted to solve a wide variety of problems. Meta-heuristic algorithms are widely recognized as one of the most practical approaches for combinatorial optimization problems. The most representative meta-heuristics include genetic algorithms, simulated annealing, tabu search and ant colony. Useful references regarding meta-heuristic methods can be found in Glover and Kochenberger (2006).

The generalized traveling salesman problem (*GTSP*) has been introduced in Laporte and Nobert (1983) and Noon and Bean (1991). The *GTSP* has several applications to location and telecommunication problems. More information on these problems and their applications can be found in Fischetti et al. (1997, 2007) and Laporte and Nobert (1983).

Several approaches were considered for solving the *GTSP*: a branch-and-cut algorithm for *Symmetric GTSP* is described and analyzed in Fischetti et al. (1997), and Noon and

Bean (1991) is given a Lagrangian-based approach for *Asymmetric GTSP*, in Snyder and Daskin (2006) is described a random-key genetic algorithm for the *GTSP*, in Renaud and Boctor (1998) it is proposed an efficient composite heuristic for the *Symmetric GTSP* etc.

The aim of this paper is to provide an exact algorithm for the *GTSP* as well as an effective meta-heuristic algorithm for the problem. The proposed meta-heuristic is a modified version of *Ant Colony System* (*ACS*). Introduced in (Maniezzo 1992; Dorigo 1992), *Ant System* is a heuristic algorithm inspired by the observation of real ant colonies. *ACS* is used to solve hard combinatorial optimization problems including the *traveling salesman problem* (*TSP*).

Definition and complexity of the GTSP

A definition of *generalized traveling salesman problem* (*TSP*) based on Laporte and Nobert (1983) and Noon and Bean (1991) follows.

Let $G = (V, E)$ be an n-node undirected graph whose edges are associated with non-negative costs. We will assume w.l.o.g. that G is a complete graph (if there is no edge between two nodes, we can add it with an infinite cost).

Let $V_1, ..., V_p$ be a partition of V into p subsets called *clusters* (i.e. $V = V_1 \cup V_2 \cup ... \cup V_p$ and $V_l \cap V_k = \emptyset$ for all $l, k \in \{1, ..., p\}$). We denote the cost of an edge $e = \{i, j\} \in E$ by c_{ij}.

The *GTSP* asks for finding a minimum-cost tour H spanning a subset of nodes such that H contains exactly one node from each cluster V_i, $i \in \{1, ..., p\}$. The problem involves two related decisions: choosing a node subset $S \subseteq V$, such that $|S \cap V_k| = 1$, for all $k = 1, ..., p$ and finding a minimum cost Hamiltonian cycle in the subgraph of G induced by S.

Such a cycle is called a *Hamiltonian cycle*. The *GTSP* is called *symmetric* if and only if the equality $c(i, j) = c(j, i)$ holds for every $i, j \in V$, where c is the cost function associated to the edges of G.

An exact algorithm for the GTSP

In this section, we present an algorithm that finds an exact solution to the *GTSP*.

Given a sequence $(V_{k_1}, ..., V_{k_p})$ in which the clusters are visited, we want to find the best feasible Hamiltonian tour H^* (w.r.t cost minimization), visiting the clusters according to the given sequence. This can be done in polynomial time by solving $|V_{k_1}|$ shortest path problems as described below.

We construct a layered network, denoted by LN, with $p + 1$ layers corresponding to the clusters $V_{k_1}, ..., V_{k_p}$ and in addition we duplicate the cluster V_{k_1}. The layered network contains all the nodes of G plus some extra nodes v' for each $v \in V_{k_1}$. There is an arc (i, j) for each $i \in V_{k_l}$ and $j \in V_{k_{l+1}}$ $(l = 1, ..., p - 1)$, with the cost c_{ij} and an arc (i, h), $i, h \in V_{k_p}$ $(l = 2, ..., p)$ with the cost c_{ih}. Moreover, there is an arc (i, j') for each $i \in V_{k_p}$ and $j' \in V_{k_1}$ with the cost $c_{ij'}$.

For any given $v \in V_{k_1}$, are considered paths from v to w', $w' \in V_{k_1}$, that visits exactly one node from each cluster $V_{k_2}, ..., V_{k_p}$, hence it gives a feasible Hamiltonian tour. Conversely, every Hamiltonian tour visiting the clusters according to the sequence $(V_{k_1}, ..., V_{k_p})$ corresponds to a path in the layered network from a certain node $v \in V_{k_1}$ to $w' \in V_{k_1}$.

Therefore the best (w.r.t cost minimization) Hamiltonian tour H^* visiting the clusters in a given sequence can be found by determining all the shortest paths from each $v \in V_{k_1}$ to each $w' \in V_{k_1}$ with the property that visits exactly one node from cluster. The overall time complexity is then $|V_{k_1}|O(m + n \log n)$, i.e. $O(nm + n \log n)$ in the worst case. We can reduce the time by choosing $|V_{k_1}|$ as the cluster with minimum cardinality. It should

be noted that the above procedure leads to an $O(nm + n\log n)$ time exact algorithm for the *GTSP*. Therefore we have established the following result:

Theorem *The above procedure provides an exact solution to the GSTP in* $O((p-1)!(nm + n\log n))$ *time, where n is the number of nodes, m is the number of edges and p is the number of clusters in the input graph.*

Clearly, the algorithm presented is an exponential time algorithm unless the number of clusters p is fixed.

Ant Colony System

Ant System proposed in Dorigo (1992) and Maniezzo (1992) is a multi-agent approach used for various combinatorial optimization problems. The algorithms were inspired by the observation of real ant colonies.

An ant can find shortest paths between food sources and a nest. While walking from food sources to the nest and vice versa, ants deposit on the ground a substance called pheromone, forming a pheromone trail. Ants can smell pheromone and, when choosing their way, they tend to choose paths marked by stronger pheromone concentrations. It has been shown that this pheromone trail following behavior employed by a colony of ants can lead to the emergence of shortest paths.

When an obstacle breaks the path ants try to get around the obstacle randomly choosing either way. If the two paths encircling the obstacle have the different length, more ants pass the shorter route on their continuous pendulum motion between the nest points in particular time interval. While each ant keeps marking its way by pheromone the shorter route attracts more pheromone concentrations and consequently more and more ants choose this route. This feedback finally leads to a stage where the entire ant colony uses the shortest path. There are many variations of the ant colony optimization applied on various classical problems.

Ant System make use of simple agents called ants which iteratively construct candidate solution to a combinatorial optimization problem. The ants solution construction is guided by pheromone trails and problem dependent heuristic information.

An individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. Each point at which an ant has to decide which solution component to add to its current partial solution is called a choice point.

After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution. Solution components which are part of better solutions or are used by many ants will receive a higher amount of pheromone and, hence, will more likely be used by the ants in future iterations of the algorithm. To avoid the search getting stuck, typically before the pheromone trails get reinforced, all pheromone trails are decreased by a factor.

Ant Colony System was developed to improve *Ant System*, making it more efficient and robust. *Ant Colony System* works as follows:

- m ants are initially positioned on n nodes chosen according to some initialization rule, for example randomly.
- Each ant builds a tour by repeatedly applying a stochastic greedy rule—the state transition rule.

- While constructing its tour, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule.
- Once all ants have terminated their tour, the amount of pheromone on edges is modified again by applying the global updating rule. As was the case in ant system, ants are guided, in building their tours by both heuristic information and by pheromone information: an edge with a high amount of pheromone is a very desirable choice.
- The pheromone updating rules are designed so that they tend to give more pheromone to edges which should be visited by ants.

The ants solutions are not guaranteed to be optimal with respect to local changes and hence may be further improved using local search methods. Based on this observation, the best performance are obtained using hybrid algorithms combining probabilistic solution construction by a colony of ants with local search algorithms as 2–3 opt, tabu-search etc.

In such hybrid algorithms, the ants can be seen as guiding the local search by constructing promising initial solutions, because ants preferably use solution components which, earlier in the search, have been contained in good locally optimal solutions.

Reinforcing Ant Colony System for GTSP

An *ACS* for the *GTSP* it is introduced. In order to enforces the construction of a valid solution used in *ACS* a new algorithm called *reinforcing Ant Colony System* (*RACS*) it is elaborated with a new pheromone rule as in Pintea and Dumitrescu (2005) and pheromone evaporation technique as in Stützle and Hoos (1997).

Let $V_k(y)$ denote the node y from the cluster V_k. The *RACS* algorithm for the *GTSP* works as follows:

- Initially the ants are placed in the nodes of the graph, choosing randomly the *clusters* and also a random node from the chosen cluster.
- At iteration $t+1$ every ant moves to a new node from an unvisited *cluster* and the parameters controlling the algorithm are updated.
- Each edge is labeled by a trail intensity. Let $\tau_{ij}(t)$ represent the trail intensity of the edge (i, j) at time t. An ant decides which node is the next move with a probability that is based on the distance to that node (i.e. cost of the edge) and the amount of trail intensity on the connecting edge. The inverse of distance from a node to the next node is known as the *visibility*, $\eta_{ij} = \frac{1}{c_{ij}}$.
- At each time unit evaporation takes place. This is to stop the intensity trails increasing unbounded. The rate evaporation is denoted by ρ, and its value is between 0 and 1. In order to stop ants visiting the same *cluster* in the same tour a tabu list is maintained. This prevents ants visiting *clusters* they have previously visited. The ant tabu list is cleared after each completed tour.
- To favor the selection of an edge that has a high pheromone value, τ, and high visibility value, η a probability function ${p^k}_{iu}$ is considered. ${J^k}_i$ are the unvisited neighbors of node i by ant k and $u \in {J^k}_i, u = V_k(y)$, being the node y from the unvisited cluster V_k. This probability function is defined as follows:

$$ {p^k}_{iu}(t) = \frac{[\tau_{iu}(t)][\eta_{iu}(t)]^{\beta}}{\Sigma_{o \in {J^k}_i}[\tau_{io}(t)][\eta_{io}(t)]^{\beta}}, \tag{1} $$

where β is a parameter used for tuning the relative importance of edge cost in selecting the next node. ${p^k}_{iu}$ is the probability of choosing $j = u$, where $u = V_k(y)$ is the next node, if $q > q_0$ (the current node is i). If $q \leq q_0$ the next node j is chosen as follows:

$$j = argmax_{u \in J_i^k}\{\tau_{iu}(t)[\eta_{iu}(t)]^{\beta}\}, \tag{2}$$

where q is a random variable uniformly distributed over [0, 1] and q_0 is a parameter similar to the temperature in simulated annealing, $0 \leq q_0 \leq 1$.

- After each transition the trail intensity is updated using the correction rule from Pintea and Dumitrescu (2005):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\frac{1}{n \cdot L^+}. \tag{3}$$

where L^+ is the cost of the best tour.

- In *ACS* only the ant that generate the best tour is allowed to *globally* update the pheromone. The global update rule is applied to the edges belonging to the *best tour*. The correction rule is

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t), \tag{4}$$

where $\Delta\tau_{ij}(t)$ is the inverse cost of the best tour.

- In order to avoid stagnation we used the pheromone evaporation technique introduced in Stützle and Hoos (1997). When the pheromone trail is over an upper bound τ_{max}, the pheromone trail is re-initialized. The pheromone evaporation is used after the global pheromone update rule.

The *RACS* algorithm computes for a given time $time_{max}$ a sub-optimal solution, the optimal solution if it is possible.

Representation and computational results

A graphic representation of *RACS* for solving *GTSP* is shown in Fig. 1. At the beginning, the ants are in their nest and will start to search food in a specific area. Assuming that each cluster has specific food and the ants are capable to recognize this, they will choose each time a different cluster. The pheromone trails will guide the ants to the shorter path, a solution of *GTSP*, as in Fig. 1.

To evaluate the performance of the proposed algorithm, the *RACS* was compared to the basic *ACS* algorithm for *GTSP* and furthermore to other heuristics from literature: *nearest neighbor* (*NN*), a composite heuristic GI^3 and a *random key-Genetic Algorithm* (Renaud and Boctor 1998; Snyder and Daskin 2006). The numerical experiments that compare *RACS* with other heuristics used problems from *TSP* library (Bixby and Reinelt 1995). *TSPLIB* provides optimal objective values for each of the problems. Several problems with Euclidean distances have been considered. The exact algorithm proposed in "An exact algorithm for the GTSP" section, is clearly outperformed by heuristics including *RACS*, because his running time is exponential, while heuristics including *RACS* are polynomial time algorithms and provide good sub-optimal solution for reasonable sizes of the problem.

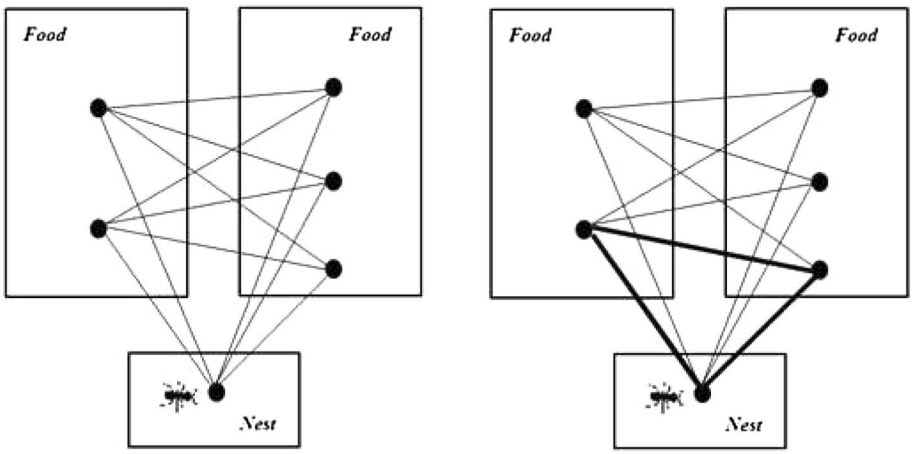

Fig. 1 A graphic representation of the *generalized traveling salesman problem* (*GTSP*) solved with an ant-based heuristic called *reinforcing Ant Colony System* (*RACS*) is illustrated. The first picture shows an ant starting from the nest to find food, going once through each cluster and returning to the nest; all the ways are initialized with the same τ_0 pheromone quantity; after several iterations performed by each ant from the nest, the solution is visible. The second picture shows a solution of *generalized traveling salesman problem* (*GTSP*) represented by the largest pheromone trail (*thick lines*); the pheromone is evaporating on all the other trails (*gray lines*)

Reinforcing Ant Colony System algorithm for the GTSP

for every edge (i,j) do $\tau_{ij}(0) = \tau_0$
  for $k = 1$ to m do
    place ant k on a randomly chosen node
    from a randomly chosen cluster
  let T^+ be the shortest tour found and L^+ its length
for $(t = 0; t < time_{max}; t++)$ do
  for $k = 1$ to m do
    build tour $T^k(t)$ by applying nc-1 times
    choose the next node j from an unvisited cluster 2
    where $J \in J_i^k$ is chosen with probability Eq 1,
    and where i is the current node
    apply the new local update rule Eq 3
  end for
  for $k = 1$ to m do
    compute the length $L^k(t)$ of the tour $T^k(t)$
  if an improved tour is found then update $T^+(t)$ and $L^+(t)$
  for every edge $(i,j) \in T^+$ do
    update pheromone trails by applying the rule Eq 4
     where $\Delta\tau_{ij}(t) = \frac{1}{L^+}$
    if $(\tau_{ij}(t) > \tau_{max})$ then $\tau_{ij}(t) = \tau_0$
  end for
end for
Print the shortest tour T^+ and its length L^+

To divide the set of nodes into subsets we used the procedure proposed in Fischetti et al. (1997). This procedure sets the number of clusters $m = [n/5]$, identifies the m farthest nodes from each other, called centers, and assigns each remaining node to its nearest center. Obviously, some real world problems may have different cluster structures, but the solution procedure presented in this paper is able to handle any cluster structure. L_{nn} is the result of *NN* algorithm. In *NN* algorithm the rule is always to go next to the nearest as-yet-unvisited location. The corresponding tour traverses the nodes in the constructed order.The initial value of all pheromone trails is τ_0.

$$\tau_0 = \frac{1}{n \cdot L_{nn}}. \tag{5}$$

For the pheromone evaporation phase, let denote the upper bound with τ_{max}.

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{L_{nn}}. \tag{6}$$

The decimal values can be treated as parameters and can be changed if it is necessary. The parameters for the algorithm are critical as in all other ant systems. Currently there is no mathematical analysis developed to give the optimal parameter in each situation. In the *ACS* and *RACS* algorithm the values of the parameters were chosen as follows: $\beta = 5, \rho = 0.5, q_0 = 0.5$.

In Table 1 are the comparative computational results for solving the *GTSP* using *ACS, RACS* and *NN, GI*[3] and *random key-Genetic Algorithm*. The columns in Table 1 are as follows.

- *Problem* The name of the test problem. The digits at the beginning of the name give the number of clusters (*nc*); those at the end give the number of nodes (*n*).
- *Opt.val.* The optimal objective value for the problem (Snyder and Daskin 2006).
- *ACS*, *RACS*, *NN*, *GI*[3], *GA* The objective value returned by the included algorithms.

Table 1 includes the best solutions in italic format. All the solutions of *ACS* and *RACS* are the average of five successively run of the algorithm, for each problem. Termination criteria for *ACS* and *RACS* is given by the $time_{max} = 10$ min. For statistics is used the *percentage relative error,* (*PER*) where the absolute error is the absolute difference between best solution and the obtain solution from Table 1.

$$PER = \frac{Absolute\ error}{best\ value} \times 100$$

The averages of *PER* are 0.71% for *ACS*, 11.50% for *NN*, 0.98% for *GI3*, 0.16% for *GA* and the best value 0.10% for *RACS*. The same for the maximal *PER* values are: 6.52% for *ACS*, 36.87% for *NN*, 5.91% for *GI3*, 2.33% for *GA* and the best value 0.87% for *RACS*. The statistics shows that *RACS* for *GTSP* comparatively performed well. It can be improved if more appropriate values for the parameters are used. Also, an efficient combination with other algorithms can potentially improve the results.

Table 1 Reinforcing Ant Colony System (RACS) versus other algorithms

Problem	Opt. val.	ACS	RACS	NN	GI^3	GA
11EIL51	174	*174*	*174*	181	*174*	*174*
14ST70	316	*316*	*316*	326	*316*	*316*
16EIL76	209	*209*	*209*	234	*209*	*209*
16PR76	64,925	*64,925*	*64,925*	76,554	*64,925*	*64,925*
20RAT99	497	*497*	*497*	551	*497*	*497*
20KROA100	9711	*9711*	*9711*	10,760	*9711*	*9711*
20KROB100	10,328	*10,328*	*10,328*	*10,328*	*10,328*	*10,328*
20KROC100	9554	*9554*	*9554*	11,025	*9554*	*9554*
20KROD100	9450	*9450*	*9450*	10,040	*9450*	*9450*
20KROE100	9523	*9523*	*9523*	9763	*9523*	*9523*
20RD100	3650	3650.4	*3650*	3966	3653	*3650*
21EIL101	249	*249*	*249*	260	250	*249*
21LIN105	8213	8215.4	*8213*	8225	*8213*	*8213*
22PR107	27,898	27,904.4	*27,898*	28,017	*27,898*	*27,898*
22PR124	36,605	36,635.4	*36,605*	38,432	36,762	*36,605*
26BIER127	72,418	72,420.2	*72,418*	83,841	76,439	*72,418*
28PR136	42,570	42,593.4	*42,570*	47,216	43,117	*42,570*
29PR144	45,886	46,033	*45,886*	46,746	*45,886*	*45,886*
30KROA150	11,018	11,029	*11,018*	11,712	*11,018*	*11,018*
30KROB150	12,196	12,203.6	*12,196*	13,387	*12,196*	*12,196*
31PR152	51,576	51,683.2	51,576.6	53,369	51,820	*51,576*
32U159	22,664	22,729.2	22,665.6	26,869	23,254	*22,664*
39RAT195	854	856.4	*854*	1048	*854*	854.2
40D198	10,557	10,575.2	10,557.6	12,038	10,620	*10,557*
40KROA200	13,406	13,466.8	13,407.2	16,415	*13,406*	*13,406*
40KROB200	13,111	13,157.8	*13,111*	17,945	*13,111*	13,113.4
45TS225	68,345	69,547.2	*68,360.6*	72,691	68,756	68,435.2
46PR226	64,007	64,289.4	64,028	68,045	*64,007*	*64,007*
53GIL262	1013	1015.8	*1015.2*	1152	1064	1016.2
53PR264	29,549	29,825	29,549.6	33,552	29,655	*29,549*
60PR299	22,615	23,039.6	22,668.2	27,229	23,119	*22,631*
64LIN318	20,765	21,738.8	*20,790.2*	24,626	21,719	20,836.2
80RD400	6361	6559.4	*6416.2*	7996	6439	6509
84FL417	9651	9766.2	9706.4	10,553	9697	*9653*
88PR439	60,099	64,017.6	60,570.6	67,428	62,215	*60,316.8*
89PCB442	21,657	22,137.8	*21,806.4*	26,756	22,936	22,134

Conclusions

The basic idea of *ACS* is that of simulating the behavior of a set of agents cooperating to solve an optimization problem by means of simple communications. The algorithm introduced to solve the *GTSP*, called *RACS*, an *ACS*-based algorithm with new correction rules. The computational results of the proposed algorithm are good and competitive in both solution quality and computational time with the existing heuristics (Renaud and Boctor 1998; Snyder and Daskin 2006). The *RACS* results can be improved with better values of parameters or using hybrid algorithms. Some disadvantages refer the multiple parameters used for the algorithm and the high hardware resources requirements.

Authors' contributions
Specifically, PCP conceived the idea of the paper. All the authors developed the simulation models. CMP executed the simulation experiments. All authors analyzed the tests results and wrote the paper. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Funding
The authors received no specific funding for the manuscript.

Publisher's Note

Received: 3 July 2017 Accepted: 31 July 2017
Published online: 07 August 2017

References

Bixby B, Reinelt G (1995) Tsplib a library of travelling salesman and related problem instances
Dahlke K (2008) Np-complete problems. Math Reference Project. Retrieved, pp 6–21
Dorigo M (1992) Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy
Dunne PE (2008) An annotated list of selected np-complete problems. COMP202, Dept. of Computer Science, University of Liverpool
Fischetti M, González JJS, Toth P (1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper Res 45(3):378–394
Fischetti M, Salazar-González J-J, Toth P (2007) The generalized traveling salesman and orienteering problems. The traveling salesman problem and its variations. Springer, Berlin, pp 609–662
Glover FW, Kochenberger GA (2006) Handbook of metaheuristics, vol 57. Springer, Berlin
Laporte G, Nobert Y (1983) Generalized travelling salesman problem through n sets of nodes: an integer programming approach. INFOR Inf Syst Oper Res 21(1):61–75
Maniezzo ACMDV (1992) Distributed optimization by ant colonies. In: Toward a practice of autonomous systems: proceedings of the first European conference on artificial life. Mit Press, Cambridge, pp 134
Noon CE, Bean JC (1991) A lagrangian based approach for the asymmetric generalized traveling salesman problem. Oper Res 39(4):623–632
Pintea C, Dumitrescu D (2005) Improving ant systems using a local updating rule. In: IEEE international symposium on symbolic and numeric algorithms for scientific computing (SYNASC 2005), 25–29 September 2005. Timisoara, Romania, pp 295–298
Renaud J, Boctor FF (1998) An efficient composite heuristic for the symmetric generalized traveling salesman problem. Eur J Oper Res 108(3):571–584
Snyder LV, Daskin MS (2006) A random-key genetic algorithm for the generalized traveling salesman problem. Eur J Oper Res 174(1):38–53
Stützle T, Hoos H (1997) Max–min ant system and local search for the traveling salesman problem. In: IEEE international conference on evolutionary computation, 1997. pp 309–314