Swarm and Evolutionary Computation 28 (2016) 78–87

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Regular Paper

A multilevel ACO approach for solving forest transportation planning problems with environmental constraints

Pengpeng Lin [a,*], Marco A. Contreras [b], Ruxin Dai [c], Jun Zhang [d]

[a] Department of Mathematics, Statistics and Computer Science, University of Wisconsin – Stout, Menomonie, WI 54751, USA
[b] Department of Forestry, University of Kentucky, Lexington, KY 40546-0073, USA
[c] Department of Computer Science and Information Systems, University of Wisconsin – River Falls, River Falls, WI 54022, USA
[d] Department of Computer Science, University of Kentucky, Lexington, KY 40506-0633, USA

ARTICLE INFO

Article history:
Received 22 October 2015
Received in revised form
18 January 2016
Accepted 19 January 2016
Available online 3 February 2016

Keywords:
Multilevel
ACO
Transportation
Graph-coarsening
Metaheuristics

ABSTRACT

This paper presents a multilevel ant colony optimization (MLACO) approach to solve constrained forest transportation planning problems (CFTPPs). A graph coarsening technique is used to coarsen a network representing the problem into a set of increasingly coarser level problems. Then, a customized ant colony optimization (ACO) algorithm is designed to solve the CFTPP from coarser to finer level problems. The parameters of the ACO algorithm are automatically configured by evaluating a parameter combination domain through each level of the problem. The solution obtained by the ACO for the coarser level problems is projected into finer level problem components, which are used to help the ACO search for finer level solutions. The MLACO was tested on 20 CFTPPs and solutions were compared to those obtained from other approaches including a mixed integer programming (MIP) solver, a parameter iterative local search (ParamILS) method, and an exhaustive ACO parameter search method. Experimental results showed that the MLACO approach was able to match solution qualities and reduce computing time significantly compared to the tested approaches.

1. Introduction

Forest transportation planning problems (FTPPs) are a special case of the fixed-charge transportation problems (FCTPs), which have received significant attention from operations research and management science [20,27]. Traditionally, FTPPs are formulated as a MIP models and solved optimally using branch-bound methods [3]. However, the computational costs of these methods increase exponentially with the problem size as FCTPs are known to be NP-hard [28,13]. To efficiently solve large-scale CFTPP, metaheuristics such as simulated annealing [2], genetic algorithm [1,16] have also been applied. For example, Contreras et al. [5] applied for the first time an ACO algorithm [7,4] to solve medium-scale FTPPs. Lin et al. [23,24] developed an improved version of the ACO algorithm to address specific FTPPs: a CFTPP and a bi-objective FTPP, respectively. Although the improved results in terms of computing time and solution quality were obtained in the experiments, solving large scale CFTPPs remains difficult because they require significantly long computing times. Moreover, the performance of the ACO algorithm is highly dependent on their parameter settings [23,10].

As a general solution strategy, multilevel schemes have been used for many years and applied to several problem areas [11,18,21,26,17] where solution quality can benefit from having a relatively high-quality initial solution that can be computed inexpensively on a lower level scale. These schemes have proven to be efficient when solving discrete NP-hard problems with a finite but exponential number of problem component combinations [30,19,29]. One recent example of using a multilevel approach to solve related transportation problems is [25] where Lin et al. developed a multilevel parameter configuration scheme and an ACO was the target algorithm configured from the coarsest to the finest level problem. Based on this previous study, we present the design, implementation, and testing of a multilevel ACO approach (MLACO) to solve large-scale CFTPPs with reduced computing times. The essential idea is to solve the original problem, which might be computationally expensive, using a set of increasingly coarser level problems on which the computational cost is cheaper. The main objective of this study is to demonstrate that, for the problem instances tested, the MLACO approach can either accelerate solution convergence rate or improve solution quality. We also examined the underlying process driving performance improvements compared to the other methods, identify advantages and limitations of the approach, and suggest how it might be applied to other optimization

* Corresponding author.
E-mail addresses: linp@uwstout.edu (P. Lin), marco.contreras@uky.edu (M.A. Contreras), ruxin.dai@uwrf.edu (R. Dai), jzhang@cs.uky.edu (J. Zhang).

http://dx.doi.org/10.1016/j.swevo.2016.01.003
2210-6502/

problems. Ultimately, the MLACO approach presented in this study can serve as a framework for solving large-scale CFTPPs and provide managers with environment-friendly road network alternatives to help them make informed decisions.

2. Preliminary

2.1. Contained forest transportation planning problem (CFTPP)

The CFTPP considered in this study is the problem of finding the set of least-cost routes from timber sale locations to designated mill destinations while reducing the negative environmental impacts associated with timber transportation [5]. Sediments expected to erode from road surfaces due to the traffic of heavy log-trucks were considered as the problem constraints. Conceptually, the CFTPP can be modeled as a network comprised of a set of nodes V and edges E representing road intersections and segments, respectively. Three attributes associated to each edge in the network are: fixed cost (*Fixed_Cost*), variable cost (*Var_Cost*), and sediment amount (*Sed*). *Fixed_Cost* is a one-time road construction cost ($) and/or maintenance cost, *Var_Cost* represents hauling cost ($) per unit of timber volume, and *Sed* (tons/year) represents the amount of sediments that are detrimental to the forest ecosystem. To formulate the CFTPP objective function, let $S=\{s_1,\ldots,s_m\}$ be the set of timber locations and $M=\{m_1,\ldots,m_n\}$ the set of mill destinations, where $S,M\subset V$. Each timber sale $s_i\in S$ has a minimum volume of timber to be delivered at a given period to a designated mill $m_j\in M$. The main objective can be defined as a cost minimization function:

$$\text{Minimize}: \sum_{E} Var\_Cost_{i,j}\times Vol_{i,j}+Fixed\_Cost_{i,j} \tag{1}$$

where $Var\_Cost_{i,j}$ is the variable cost, $Fixed\_Cost_{i,j}$ the fixed cost, and $Vol_{i,j}$ the total timber volume transported from node i to j ($Vol_{i,j}=0$ if the road segment ij is not used). Also, the total timber volumes arriving at mills must agree with the total timber volumes shipped out from the timber sales:

$$\sum_{i=1}^{m} Vol_{s_i}=\sum_{j=1}^{n} Vol_{m_j} \tag{2}$$

and the amount of sediment eroding from the entire transportation network must not exceed a maximum allowable value:

$$Constraint: \sum_{E} Sed_{i,j}\le Sed_{max}. \tag{3}$$

where Sed_{max} is the maximum sediment threshold. The equality (2) and the inequality (3) are the constrains in addition to minimizing the objective function (1) to determine the optimal solution for the CFTPP. A detailed description of CFTPPs can be found in [5,23,24].

2.2. Ant colony optimization

ACO was developed in the mid 1990s to solve the traveling salesman problem [9,8]. The algorithm was inspired by ant foraging behavior. When searching for food, ants walking to and from a food source deposit a substance called pheromone on the ground. Other ants can perceive the presence of the pheromone and tend to follow paths where pheromone concentrations are higher.

In the ACO algorithm to find minimum routes, a set of artificial ants are placed at origin locations and move through adjacent locations one at a time towards the destinations. Guided by the pheromone values, artificial ants construct routes simultaneously. Let C be a set of all possible locations, an ant placed at location x chooses what location y to visit next according to a transition probability:

$$P_{x,y}^{t}=\begin{cases}\dfrac{\mathcal{T}_{x,y}^{\alpha}\times\eta_{x,y}^{\beta}}{\sum_{k\in Nbr}\mathcal{T}_{x,k}^{\alpha}\times\eta_{x,k}^{\beta}} & \text{if } y\in cities\\ 0 & \text{Otherwise}\end{cases}$$

where $x,y\in C$, $P_{x,y}^{t}$ is the probability of ant t moving from x to y, $k\in Nbr$ represents one of unvisited locations adjacent to x, τ is the pheromone intensity on the path connecting two locations, η is the visibility (typically calculated as the inverse to the distance between the two locations) α and β are positive parameters that control the relative importance of pheromone intensity versus visibility. The pheromone intensity τ is updated iteratively using the following formula:

$$\mathcal{T}_{x,y}\leftarrow\rho\times\mathcal{T}_{x,y}+\Delta\mathcal{T}_{x,y},$$

where ρ is the pheromone persistence rate and $\Delta\tau_{x,y}$ is the amount of pheromone to be added to path (x,y). For a more detailed description of ACO algorithm, see [7].

2.3. Multilevel scheme

Typically, a multilevel scheme solves a large problem using a set of increasingly smaller problems through a sequence of solution refinements [25]. These smaller problems are obtained by successively applying a coarsening process to the original problem. As a result, a hierarchy of coarser problems are generated where a given coarser level problem is always smaller than its finer level problem. The solution obtained for a given coarser level problem in the solution refinement process is projected into the finer level problem components which are then used to help search for the finer level solution. The process is illustrated in Fig. 1 where a finer problem is coarsened into a coarser problem. After the ACO algorithm is applied to a given coarser problem, the solution is interpolated into a set of finer level components that can help the ACO algorithm find good solutions for the finer level problem.

For clarity, we define the following terms:

Coarser/finer level problems: a set of increasingly coarser level problems $\Pi=\{\Pi_0,\Pi_1,\ldots,\Pi_N\}$ where Π_0 is the original

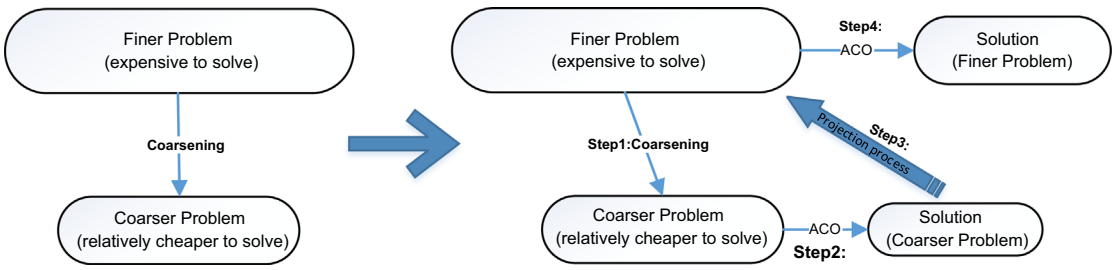

Fig. 1. Diagram illustrating a multilevel scheme at its simplest form (only two levels), where the finer level problem is coarsened into a coarser level problem (left-hand side). The ACO algorithm solves the coarser level problem first and the obtained solution is used to help find high-quality solutions for the finer level problem (right-hand side).

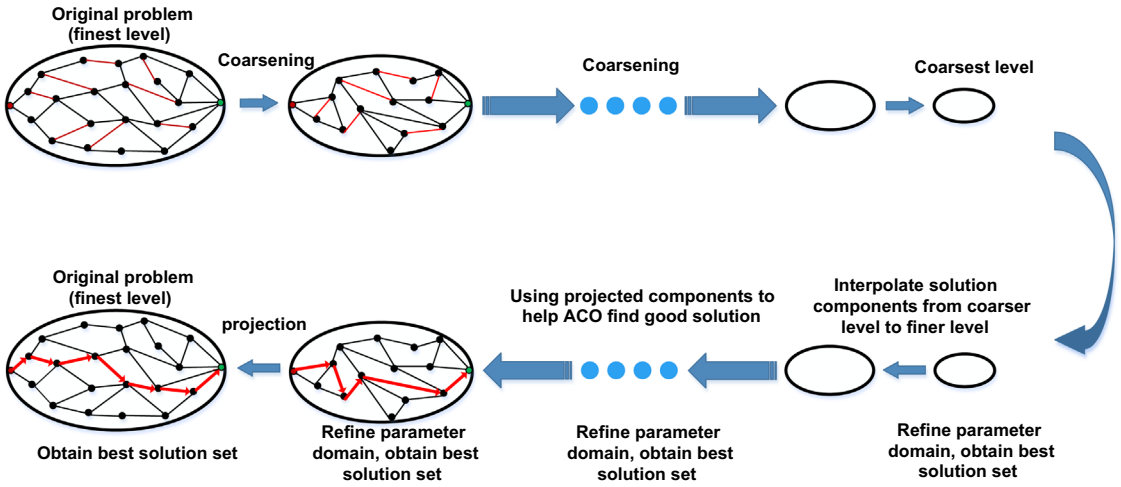

Fig. 2. The upper left corner of the diagram shows the original problem network where the objective is to find a route from an origin (red node) to a destination (green node). The matching edges (red edges) are contracted to produce coarser level problems. The original problem is coarsened into a set of increasingly coarser problems. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

problem and Π_N is the coarsest level problem. If a problem $\Pi_i \in \Pi$ is referred to as a coarser level problem, then the problem $\Pi_{i-1} \in \Pi$ is referred to as its finer level problem and vice versa.

Projected components: for a coarser level problem Π_i and its finer level problem Π_{i-1}, the projected components refer to the finer level problem components obtained by interpolating the solution found for the coarser level problem.

Projection process: for a coarser level problem Π_i and its finer level problem Π_{i-1}, the projection process interpolates the coarser level solution to obtain the projected components.

3. Multilevel ACO approach

The MLACO approach presented in this paper is designed based on the multilevel scheme in which coarser level problems are produced by using the graph coarsening algorithm proposed in [12]. It works as follows: starting from the original problem, it finds a set of matching edges in which no two edges are incident to the same node. It then collapses the matching edges and aggregates the incident nodes to produce a coarser level problem with fewer number of edges and nodes. Next, the same process is applied to the coarser level problem to generate the next coarser level problem with fewer edges and nodes than the previous one. The coarsening algorithm repeats this process until a threshold, such as number of coarser level problems to be produced, is met. As a result, a sequence of coarser level problems is obtained. For clarity, in the rest of the paper, the nodes on the collapsed matching edges are referred to as "matching nodes", the coarser level nodes obtained by aggregating the matching nodes are referred to as "aggregated nodes", and the coarser level nodes that are the same as its finer level are referred to as "unaggregated nodes".

The next step of the MLACO approach is to apply the ACO algorithm to solve the original problem in a reverse order by applying it first to the coarsest level problem. The obtained solution for the coarsest level is interpolated into projected components for the second coarsest level which are given larger amount of pheromone values than other problem components during the solution search process of the ACO algorithm. Next, the ACO is applied to the second coarsest level problem (or the corresponding finer level problem) to search for the best solution with preference given to the projected components. The obtained solution for the second coarsest level problem is then interpolated into projected components which are used to help the ACO algorithm search for the best solution for the third coarsest level problem. This process is repeated at each coarser level problem until the original problem is solved. The rationale behind this design is that most (if not all) of projected components are assumed to be optimal solution components of the next coarsest level problem. Guided by these problem components at the beginning of the search process, the ACO algorithm is expected to converge towards the optimal (or high-quality) solution much faster.

In addition to the solution search and refinement process, we also adopted the multilevel parameter configuration approach from [25] to achieve the maximum ACO performance. In the MLACO approach, the parameter settings of the ACO algorithm are refined by selecting high quality parameter values from a predefined parameter combination domain that includes all possible parameter combinations with a given value interval for each configured parameter. The selection decision is based on the quality of the solution obtained at each iteration. As a result, low quality parameter values from the domain are discarded. This parameter refinement process is applied from coarsest level problem to the finest level (original) problem within the solution search process. As ACO algorithm performance is highly sensitive to its parameter settings [23,10] because of its stochastic nature, integrating this approach into the MLACO design is expected to increase the change for finding the optimal solutions. The MLACO approach is illustrated in Fig. 2.

3.1. Underlying guidedACO

The ACO algorithm (Algorithm 1) used in the MLACO approach is referred to as GuidedACO as its search process is guided by the projected components. The projected components are initialized (in Algorithm 2) with larger amounts of pheromone values to increase their probabilities of being selected for constructing solutions. Specifically in this study, pheromone values for the projected components are set to be the inverse of the normalized heuristic value ($\eta_{C_i}/\max \eta$) multiplied by an updating factor (Update_Factor – discussed later). This setting ensures that the projected components with smaller heuristic values (such as costs) receive a larger amount of pheromone and vice versa.

In trial experiments, we found that the GuidedACO can find several solutions with the same quality (same objective function value and same total sediment value), indicating the existence of multiple optimal solutions. This is especially true for constrained optimization problems where two solutions can be equivalent. For instance, two equivalent solutions might have either the same objective value but

different constraint values, or same objective and constraint value but different solution components. The former case is expected to occur more frequently than the latter because seldom two solutions with different components have identical objective and constraint values. Consequently, equivalent solutions in the CFTPP are defined as follows:.

Definition 1. Two feasible solutions S_i and S_j are equivalent, denoted as $S_i \| S_j$, if their objective function values are equal: $Obj(S_i) = Obj(S_j)$, and if there is at least one solution component C such that $C \in S_i \wedge C \notin S_j$ or vise versa.

The solution search process (**SearchProcess** in Algorithm 1) is comprised of two stages. During the first stage, it starts with a solution search where the objective is to determine the existence of feasible solutions. When a feasible solution is found, the second stage takes place where the objective changes to find the best feasible solution. During the first stage, the algorithm is likely to find a number of infeasible solutions before a feasible solution is found. Thus, pheromone values are updated only when an infeasible solution with the feasible condition closer than all previously obtained infeasible solutions is found. At the same time, a counter (that tracks the number of consecutive iterations that have not received pheromone update) is reset to zero to allow more iterations and improve solution quality. If the GuidedACO cannot find a feasible solution during the first stage, the search process is stopped. The transition probability in the first stage is defined in Formula (4):

$$P_t(c_{i,j}) = \frac{(\mathcal{T}_{i,j})^{\alpha} \times (Sed_{i,j}^{-1})^{\beta}}{\sum_{i,k \in Nbr}(\mathcal{T})^{\alpha} \times (Sed_{i,j}^{-1})^{\beta}} \quad (4)$$

where t is the iteration number, c is the solution component and Nbr represents all available adjacent components.

Algorithm 1. GuidedACO ($\Pi, S_{component}, \theta$).

```
Initialization(BestSolus ⟸ φ; CurBestSolu ⟸ φ.)
Output(BestSolus)// equivalent best solution set
begin
    Set_Phero(Π,S_component)
    while stop conditions are not satisfied do
        if First Stage then
            CurBestSolu ⟸ run ACO SearchProcess with θ;
            if CurBestSolu is better solution then
                Phero_Update(Π,CurBestSolu);
            end
        end
        if Second Stage then
            CurBestSolu ⟸ run ACO SearchProcess with θ;
            if CurBestSolu is an equivalent best solution then
                Add CurBestSolu to BestSolus;
                Phero_Update(Π,CurBestSolu);
            end
            else if CurBestSolu is better than existing solutions in BestSolus then
                Update BestSolus according to CurBestSolu;
                Phero_Update(Π,CurBestSolu);
            end
        end
    end
end
```

Algorithm 2. Set_Phero ($\Pi, S_{component}$).

```
begin
    foreach C_i ∈ Π do
        τ_{C_i} ⟸ Initial_Pheromone
        if C_i ∈ S_component then
            τ_{C_i} ⟸ (η_{C_i} / max(η))^{-1} × Update_Factor × τ_{C_i}
        end
    end
end
```

During the second stage, the algorithm is expected to find equivalent feasible solutions. In the case that an obtained solution is equivalent to the current best solution, the GuidedACO includes it in a set (**BestSolus** in Algorithm 1) that stores all equivalent best solutions found from previous algorithm iterations. On the other

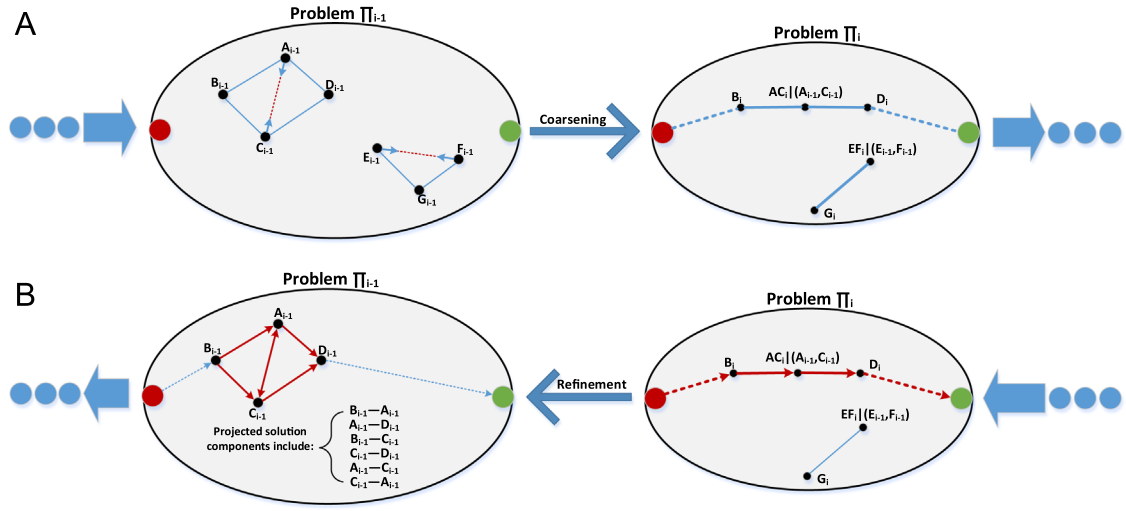

Fig. 3. Diagram illustrating the problem coarsening phase (A), and the projection process in the solution refinement phase (B). In the problem coarsening phase, the problem Π_{i-1} is coarsened to obtain coarser level problem Π_i. In the solution refinement phase, after interpolated the solution components e_{B_i,AC_i} and e_{AC_i,D_i} in Π_i, the possible solution combinations that connect from node B_{i-1} to node D_{i-1} are $(B_{i-1}\rightarrow A_{i-1}\rightarrow D_{i-1})$, $(B_{i-1}\rightarrow C_{i-1}\rightarrow D_{i-1})$, $(B_{i-1}\rightarrow A_{i-1}\rightarrow C_{i-1}\rightarrow D_{i-1})$, $(B_{i-1}\rightarrow C_{i-1}\rightarrow A_{i-1}\rightarrow D_{i-1})$. The projected components are edges appear in all the possible solution combinations: $e_{B_{i-1},A_{i-1}}$, $e_{A_{i-1},D_{i-1}}$, $e_{B_{i-1},C_{i-1}}$, $e_{C_{i-1},D_{i-1}}$, $e_{A_{i-1},C_{i-1}}$, $e_{C_{i-1},A_{i-1}}$.

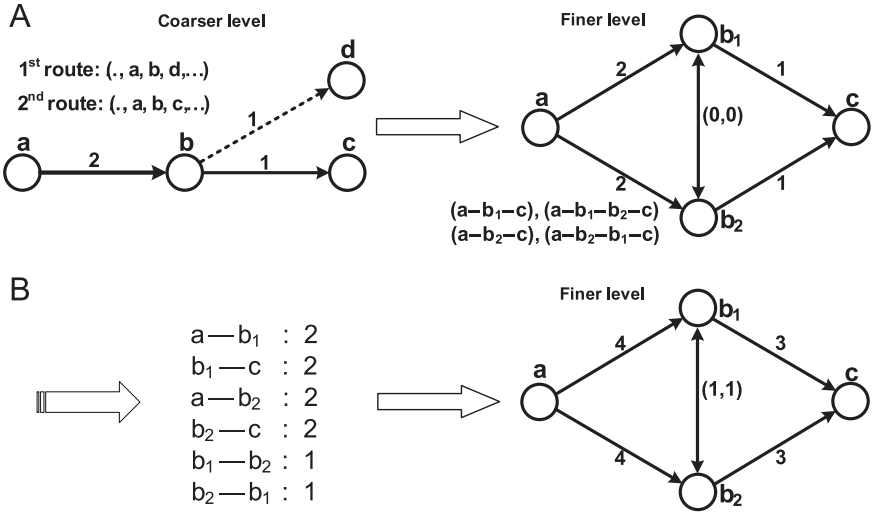

Fig. 4. Calculating update factor values for projected solution components.

hand, if the obtained solution is better than the current best solution, it becomes new current best solution and those solutions previously stored are removed from the set. The transition probability in this stage is defined in Formula (5):

$$P_t(c_{i,j}) = \frac{(\mathcal{T}_{i,j})^{\alpha} \times [\lambda \times NFCost_{i,j}^{-1} + (1-\lambda) \times Sed_{i,j}^{-1}]^{\beta}}{\sum_{i,k \in Nbr} (\mathcal{T}_{i,k})^{\alpha} \times [\lambda \times NFCost_{i,k}^{-1} + (1-\lambda) \times (Sed_{i,k}^{-1})]^{\beta}} \quad (5)$$

in which $NFCost_{i,j}$ is the unit cost (summation of the fixed and variable costs per timber volume) for the edge (i,j) calculated as

$$NFCost_{i,j} = \frac{Fixed\_Cost_{ij}}{\sum_{q \in Q} Vol_q} + Var\_Cost_{ij} \quad (6)$$

where Q represents all timber sales routes that use the component $c_{i,j}$, $\sum_{q \in Q} Vol_q$ is total timber volume transported through the component $c_{i,j}$, and the parameter λ is a weight used to balance the importance of cost and sediment values.

Algorithm 3. Phero_Update(Π, CurrentBestSolution).

```
begin
    foreach C_i ∈ Π do
        if C_i ∈ CurBestSolu then
            τ_{C_i} ⟸ τ_{C_i} + Δτ, where Δτ is a small value
        end
        else
            τ_{C_i} ⟸ τ_{C_i} × ρ, where 0 < ρ ≤ 1
        end
    end
end
```

Pheromone values in both stages are updated using the current best solution for every iteration. The GuidedACO increases pheromone values for the current best solution components by a small

amount $\Delta\tau$ and decreases pheromone values for all other problem components by $(1-\rho)$ (Algorithm 3). The GuidedACO stops searching for better solutions when the set of equivalent best solutions reaches stagnation.

3.2. MLACO

The MLACO approach (Algorithm 4) uses the GuidedACO to solve the CFTPP through a process of solution and parameter refinement over a set of increasingly coarser level problems. It consists of a problem coarsening phase and a solution refinement phase (as illustrated in Fig. 2). In the problem coarsening phase, coarser level problems are produced using the graph coarsening algorithm which also tracks coarsening information that includes contracted edges and aggregated weights. For example, let a problem Π_0 be coarsened into coarser level problems $\{\Pi_1, \Pi_2, \ldots, \Pi_N\}$ and $k = \{k_0, k_1, \ldots, k_{N-1}\}$ be the corresponding coarsening information for all coarser level problems, $k_i \in k$ is defined as a set of triplets:

$$k_i = \left\{ k_i \middle| \left(v_{a_1}^{\Pi_{i-1}}, v_{b_1}^{\Pi_{i-1}}, v_{c_1}^{\Pi_i}\right), \left(v_{a_2}^{\Pi_{i-1}}, v_{b_2}^{\Pi_{i-1}}, v_{c_2}^{\Pi_i}\right), \ldots, \left(v_{a_d}^{\Pi_{i-1}}, v_{b_d}^{\Pi_{i-1}}, v_{c_d}^{\Pi_i}\right) \right\} \quad (7)$$

where d is the number of the aggregated nodes and each triplet contains two finer level matching nodes v_a, v_b and the resulted coarser level aggregated node v_c.

Algorithm 4. MLACO.

```
Input(Increasingly coarser problems D ⇔ Π_0, Π_1, ..., Π_N where Π_0 is the original problem;
Coarse information k = k_0, k_1, ..., k_{N-1}; A parameter combination domain Θ.)
begin
    S ⟸ φ // S is an empty solution set
    Best_Solu ⟸ Evaluate(Θ, Π_N, S)
    for i = N − 1 to 0 do
        S ⟸ Projection(Π_i, Π_{i−1}, k_i, Best_Solu)
        Best_Solu ⟸ Evaluate(Θ, Π_i, S)
    end
    Return(Best_Solu)
end

/* Obtain solution with given problems                                          */
Procedure: Evaluate(Θ, Π, S)
begin
    foreach θ ∈ Θ do
        LocalBestSolu ⟸ running GuidedACO(Π, S, θ)
        if solutions in LocalBestSolu better than that of GlobalBestSolu then
            GlobalBestSolu ⟸ φ
            GlobalBestSolu ⟸ LocalBestSolu
        end
        else if solutions in LocalBestSolu equivalent to that of GlobalBestSolu then
            GlobalBestSolu ⟸ GlobalBestSolu + LocalBestSolu
        end
        if solutions in LocalBestSolu have very low qualities then
            remove θ from Θ
        end
    end
    Return(GlobalBestSolu)
end

/* Obtain the projected components                                              */
Procedure: Projection(Π_i, Π_{i−1}, k_i, Best_Solu_{Π_i})
begin
    Projected_Comps ⟸ φ foreach solution S_{Π_i} ∈ Best_Solu_{Π_i} do
        foreach component C_{Π_i} ∈ S_{Π_i} do
            Obtain projected solution components SC_{Π_{i−1}} ⊂ Π_{i−1} using k_i
            foreach component C_{Π_{i−1}} ∈ SC_{Π_{i−1}} do
                Calculate its Update_Factor
                add C_{Π_{i−1}} to Projected_Comps
            end
        end
    end
    Return(Projected_Comps)
end
```

In the solution refinement phase, the best found solution for a coarser level problem is interpolated with the coarsening information to produce projected components (Projection procedure in Algorithm 4) which are the finer level problem components contracted and aggregated to produce the coarser level solution components (Fig. 3). The GuidedACO uses the projected components to construct solutions for the finer level problem and iteratively refines and improves the solution quality. The obtained best solution for the finer level problem, in turn, is interpolated again with the coarsening information to produce a new set of projected components that are used in the GuidedACO to solve the next finer level problem. The MLACO repeats the same steps subsequently from coarser to finer level until the finest level problem (original) is solved.

While the projected components are used as initial solution and refined iteratively to obtain better quality solutions, the MLACO also attempts to achieve maximum performance by automatically selecting high-quality parameter combinations (Evaluate procedure in Algorithm 4) from coarser to finer level problems. When a new solution is obtained, it is compared to the solutions in the best solution set obtained from previous iterations. If the new solution is equivalent or better, it is included in the best solution set and the associated parameter combination is considered high-quality. Otherwise, it is discarded along with the parameter combination used. By identifying and discarding low-quality parameters, the overall computing time for configuring the GuidedACO is reduced because of the fewer number of parameter combinations requiring evaluation at the final level problem.

3.3. Calculating update factor

When setting pheromone values in a finer level problem, different projected components are given different pheromone amounts. Each projected component is associated with an update factor that reflects the importance of the projected component based on the interpolated coarser level solutions. Depending on the values of the update factors, the amount of pheromone is assigned by giving a larger amount of pheromone to projected components associated with larger update factors and vice versa (Algorithm 2).

For a coarser level problem, after the GuidedACO is applied, a number of equivalent solutions are found. Each of the equivalent solutions is then interpolated to obtain a set of projected components. As one solution component can exist in several equivalent coarser level solutions, a projected component might also be obtained more than once. The number of times a projected component is obtained by the projection process indicates how frequently the corresponding coarser level component is used as the solution component and can be considered as a metric to calculate the associated update factor.

An example of the calculation of the update factor is illustrated in Fig. 4 where the solution set of a coarser level problem contains two routes that pass through edges $e_{a,b}$, $e_{b,c}$, and $e_{b,d}$ (1^{st} and 2^{nd} routes in left picture in Fig. 4A). Each edge is associated with a number indicating the number of times the edge is used in the solution set (i.e., the number for the edge $e_{a,b}$ is two since it is used in both routes 1 and 2). After the projection process, nodes a and c stay the same and node b is replaced with finer level nodes b_1 and b_2 to produce the projected components because b is an aggregated node (right picture in Fig. 4A). If a projected component is incident to one or two unaggregated nodes (such as e_{a,b_1} and $e_{b_1,c}$), it is assigned the number associated with the coarser level solution component incident to the same nodes. Otherwise, zero is assigned to the projected component incident only to the finer level nodes (i.e., e_{b_1,b_2} and e_{b_2,b_1}). As the coarser level solution route

Table 1
Summary of experiment setup, parameter setting, implementation methods and running environment considered for comparisons.

Method	Description	Parameter Settings	Implementation	OS
Exhaustive parameter search (EPS)	GuidedACO runs 10 times for every parameter setting to obtain the best solution.	$\alpha \in [0.05, 0.1, ..., 0.95, 1]$, $\beta \in [0.05, 0.1, ..., 0.95, 1]$, $\rho \in [0.05, 0.1, ..., 0.95, 1]$, $\Delta\tau = \tau_{G_i} \times 0.0001$, $\lvert\Theta\rvert = 8000$, $\lambda = 0.7$, *Counter Threshold* = 10000.	Divide Θ into 20 partitions and use 20 processors to run GuidedACO with each parameter combination partition simultaneously	C++, Linux
ParamILS	ParamILS configures parameter setting of GuidedACO to obtain best solution	Same settings for GuidedACO, Number of iterations: 100, Local search $r = 10$, Perturbation $s = 3$, Initial Setting: ($\alpha = 0.5, \beta = 0.5, \rho = 0.5$)	Parameter settings of GuidedACO is sequentially configured and ParamILS stops after 100 iterations. Best solution is obtained during configuration	C++, Linux
MIP	CFTPP formulated into linear programming model and solved using CPLEX 12.5	Thread Number: 1, Run Time: 864,000 s, Others: default settings	Sequentially solve all problems	Java and Cplex, Linux
MLACO		Same settings for GuidedACO, Four level problems: Level-0 (original problem), Level-1 (first coarser), Level-2 (second coarser), Level-3 (third coarser)		C++, Linux

goes from nodes a to c, the projected components are expected to also connect these two nodes, which results in four possible paths: $(a \rightarrow b_1 \rightarrow c)$, $(a \rightarrow b_2 \rightarrow c)$, $(a \rightarrow b_1 \rightarrow b_2 \rightarrow c)$, $(a \rightarrow b_2 \rightarrow b_1 \rightarrow c)$ (left picture in Fig. 4B). Counting the solution component occurrences in the four paths, e_{a,b_1}, $e_{b_1,c}$, e_{a,b_2}, $e_{b_2,c}$ appear twice and e_{b_1,b_2}, e_{b_2,b_1} appear once. Then the update factor values are calculated by adding these occurrences to the existing assigned numbers (right picture in Fig. 4B).

4. Experiment setup

The algorithms and procedures presented in this study were implemented using C++ and Java and uploaded to the Lipscomb High Performance Computing Cluster (HPC) supported and maintained by the University of Kentucky Center for Computational Science. All programs were executed on the computing nodes of HPC: Dual Intel E5-2670 of 8 Cores at 2.6 GHz with 64 GB of 1600 MHz RAM and Linux Red Hat OS. We compared the performance of the MLACO approach with other three methods: an exhaustive parameter search (EPS) method, a ParamILS method [15], and the MIP solver (Table 1).

The configured parameters included α, β, ρ which values were confined to [0,1] with a pace of 0.05, resulting in a 8000 parameter combination domain (excluding zero parameter values). The EPS method run the GuidedACO 10 times for each parameter combination in the domain and selected the feasible solution with the best objective function value from the 10 runs. Because EPS methods can often require long computing time, we divided the parameter combination domain into 20 partitions and applied the EPS method to all partitions simultaneously. The computing time of the EPS method was calculated by adding the times required for all partitions. The ParamILS method configured parameter settings for the GuidedACO based on solution quality. It iteratively permuted parameter values and run the GuidedACO until it could not find a better quality solution. The MIP formulation of the CFTPP presented in [23] was implemented using Java and solved using the CPLEX 12.5 Callable Library (ILOG Inc. 2007) with default parameter setting [6]. Because MIP solvers can also take impractically long computing time, we set its maximum running time to 864,000 s (10 days), after which CPLEX was forced to stop and report the best solution found.

The MLACO approach used a set of three increasingly coarser level problems: Level-1, Level-2 and Level-3 (Table 1). After initial test runs, three coarsening levels were selected to balance solution quality and computation time as well as preserving the properties of the original problem. This resulted in the coarsest level problem to be about one eighth of the size of the original problem (problem size was reduced by about one half from a given level to its coarser level).

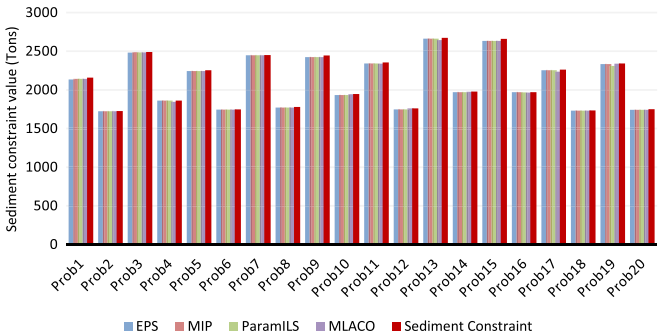

Fig. 5. Sediment value (tons) associated with the best solution found by the four methods for each problem tested.

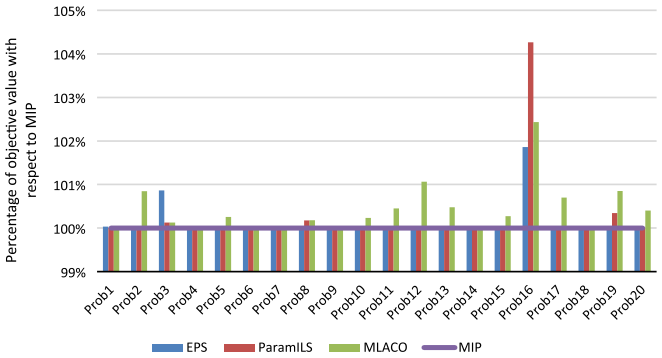

Fig. 6. Percentage of the objective function value found by the four methods for each problem tested. The objective function values of other methods are divided by those obtained by MIP to calculate the percentages and MIP results are marked at 100% level (purple line). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 2
Statistical test (univariate test with Tukey post hoc) of objective function values and computing time for the tested methods. The EPS method was not included in the statistical test for computing time as the differences between it and other methods were evidently significant shown in Fig. 7 and Table 3.

Objective function value			**Computing time**		
(I) Method	(J) Method	p-value	(I) Method	(J) Method	p-value
EPS	MIP	0.707	MIP	MLACO	< 0.0001
	MLACO	0.969		ParamILS	< 0.0001
	ParamILS	0.845			
MIP	Exhaustive	0.707	MLACO	MIP	< 0.0001
	MLACO	0.43		ParamILS	< 0.0001
	ParamILS	0.243			
MLACO	Exhaustive	0.969	ParamILS	MIP	< 0.0001
	MIP	0.43		MLACO	< 0.0001
	ParamILS	0.983			
ParamILS	Exhaustive	0.845			
	MIP	0.243			
	MLACO	0.983			

Table 3
Summary of computing time required to solve all problems for the Exhaustive, ParamILS, MIP, and MLACO.

(Days)	**Exhaustive**	**ParamILS**	**MIP**	**MLACO**
Min	137	6	0	0
Max	227	19	10	4
Median	183	11	10	1
Average	184.352	12.274	6.009	1.288
Std Dev	26.079	3.280	4.888	1.053

For the experimental data, we used 10 network instances containing the 20 CFTPPs used in Lin, et al., in [23]. These problems were designed as medium-scale, grid-shaped hypothetical network with 500 road segments, 25 timber sale locations, and a mill destination. The hypothetical grid-shaped network was used because it resembles real-world FTPPs, thus providing a good testing case for algorithm performance. In addition, the medium-scale size allows solving the instances a large number of times within reasonable time to conduct the parameter search. All problem instances can be downloaded from [22].

5. Experimental results

The experiments were conducted to test performance in terms of solution quality and computing time. All methods were able to find

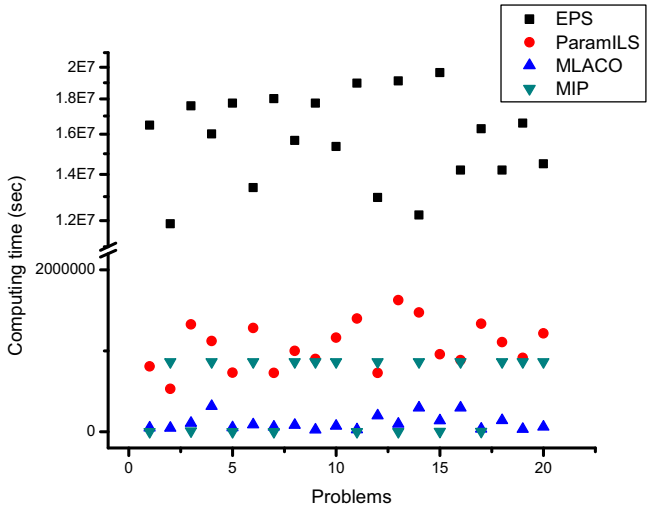

Fig. 7. Computing time (s) required by the four methods to obtain the best found solution for each problem tested.

feasible solutions for all problems (obtained sediment amounts below the constraint values as shown in Fig. 5). As MIP solver CPLEX can solve CFTPPs optimally, its solutions were used to benchmark solution qualities achieved by the EPS, ParamILS and MLACO methods (Fig. 6). These three approximation methods were able to match MIP solutions for most problems, except for problems 3, 16, 19 and 20 where solution qualities were slightly worse. In the worst case (problem 16), the MLACO approach was able to outperform the ParamILS method and was slightly worse than the EPS method. In addition, univariate statistical tests (Table 2) show that there were no significant differences of solution quality between MIP and any other tested methods, indicating that on average the three approximation methods are expected to obtain near-optimal solutions for the tested problems. This also indicates that the MLACO approach was able to self-configure properly and obtained competitive high-quality solutions compared to the other methods.

In terms of computing time, results show significant differences among the tested methods (Table 3). For the test cases that required the longest running time for each method, the EPS method spent 227 days, ParamILS 19 days, MIP 10 days and MLACO 4 days, while for the cases the required the least amount of time, the EPS method spent 137 days, ParamILS 6 days, and both MIP and MLACO less than one day. For the average computing time required, the EPS method spent the longest (184.35 days) compared to other methods, followed by the ParamILS that spent 12.27 days (about 6.6%), the MIP solver 6 days (3.2%), and the MLACO approach only spent 1.28 days. Detailed computing time comparisons including testing problems are shown in Fig. 7.

Although MIP solvers typically require relatively long computing time, on average the CPLEX spent less time to solve all problems compared with the EPS and ParamILS methods. This resulted because the MIP solver quickly found optimal solutions for some problems (Fig. 7), which reduced the average computing time. Compared to all other methods, the MLACO approach required significantly less amount of computing time. The variation in computing time among problems was largest for the EPS method (Std Dev: 26.7 days) followed by the MIP solver (Std Dev: 4.8 days) and ParamILS method (Std Dev: 3.2 days). In contrast, the MLACO approach showed the most stable computing times (Std Dev: 1.05 days). Also, computing time for the MIP solver was highly skewed by problems that were solved very quickly (Fig. 7). For the remaining problems, the MIP solver spent exactly 10 days indicating it was not able to find the optimal solutions and was forced to stop after the maximum running time. As suggested on

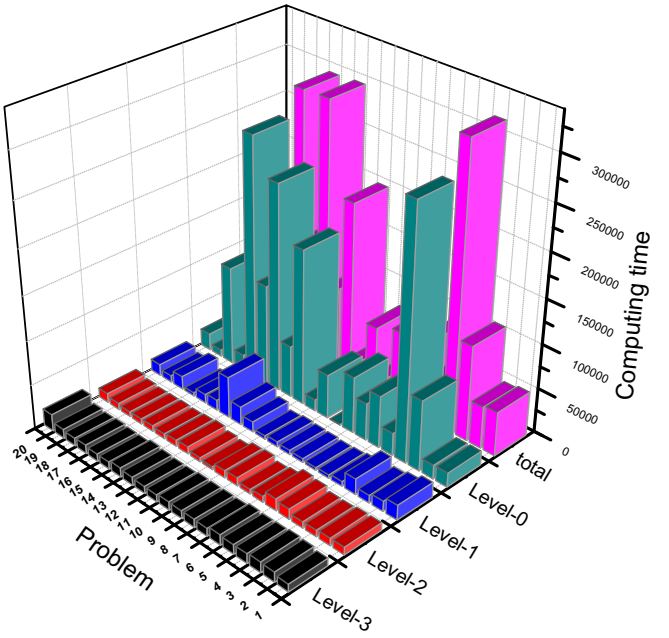

Fig. 8. Computing time (s) required by the MLACO approach to solve each problem by coarsening level.

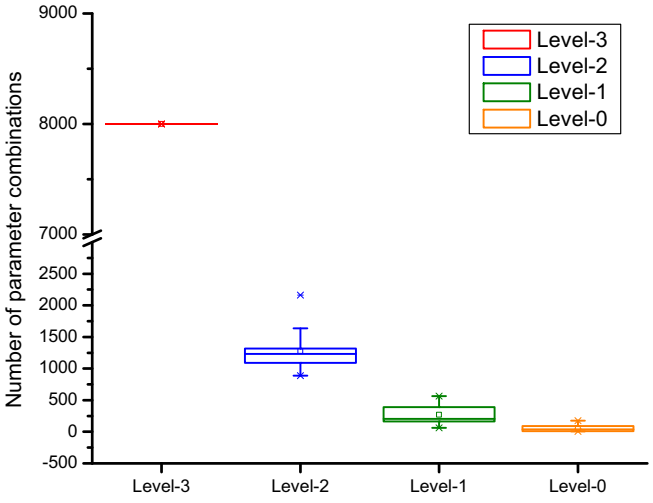

Fig. 9. Parameter combination domain size changes at each level of problems for MLACO.

the CPLEX reference manual [14], MIP performance is affected by its parameter setting. However, it is impractical to configure all 135 MIP parameters driving the search process.

Next, we analyzed the performance of the MLACO approach by examining the parameter domain size and computing time required for solving each level of the coarser problems by the GuidedACO. The computing time is related to the number of parameter combinations evaluated. For example, the minimum computing time required for all problems at each level was largest for Level-3 and smallest for Level-0, 8000 and 6 respectively (Table 4), indicating that quicker computing time was achieved for a smaller parameter domain size. On the other hand, the maximum computing time required for all problems at each level was largest for Level-0 (280,291 s) with the least number (176) of parameter combinations to evaluate and was relatively small for Level-3 (18,796 s) and Level-2 (13,725 s) for which larger numbers of parameter combinations were evaluated (8000 and 2162 parameter combinations). This indicates that the computing time for solving a coarser level problem was also directly affected by the problem size and complexity. We can also observe that the computing times

Table 4
Computing time required by the MLACO approach and parameter combination domain size for each coarse level problem. Level-0 denotes for the original problem.

	Computing time (s)				Size of parameter combination domain			
	Level-3	Level-2	Level-1	Level-0	Level-3	Level-2	Level-1	Level-0
Min	7978	3436	3143	512	8000	889	63	6
Max	18,796	13,725	46,478	280,291	8000	2162	564	176
Median	9310	7301	9284	54,112	8000	1239	235	37
Average	10,076	7609	11,243	82,367	8000	1275	270	58
Std Dev	2337.848	2339.627	9542.240	86,383.311	0	275.652	142.449	53.721

required for the last level coarser problem (Level-0) for all CFTPPs were significantly more than the computing times required for other coarser level problems (Fig. 8), and the computing time differences between Level-1, Level-2 and Level-3 were considerably smaller. This suggests that there is a need to design a scheme to determine number of coarser level problems to be used in the MLACO for the future study. Lastly, the size of parameter combination domain decreased significantly (Fig. 9) from Level-3 (8000 on average) to Level-2 (1275) and continued to decrease to Level-1 (270) and to Level-0 (58). This result shows that as problem complexity increased, the number of high-quality parameter combinations for the problem was reduced, which might indicate that the performance of the MLACO approach was less sensitive to its parameter setting for a coarser level problem than that for a finer level problem.

6. Conclusion

In this study, a novel multilevel ACO approach (MLACO) has been developed to solve the CFTPP. The approach was designed to use a set of increasingly coarser level graphs to condense global level information for efficient handling of large-scale applications by the ACO algorithm. It also uses parameter domain of coarser levels to restrict the search on the domain of finer levels to quickly find parameters yielding high-quality solutions. Moreover, it uses paths on coarser levels as initial solutions of the finer levels for a more rapid ACO algorithm convergence to optimal or near-optimal solutions and avoid initial random solution search on high-cost paths. The salient feature of the developed MLACO approach is that these beneficial features can be readily extended to other FCTP and other optimization problems with underlying graph structures.

The developed approach was able to find near-optimal solutions with a significant reduction of computing time for the 20 CFTPP instances, which had similar topology and complexity as real-world problems. These results indicate the great potential of the MLACO approach to serve as a generalized framework to solve large-scale, real-world transportation problems. Lastly, in the case of FTPP applications, by the use of constraints, it allows the incorporation of increasingly important social and environmental aspects into transportation planning to provide managers with economically efficient and environmentally sound transportation alternatives.

References

[1] K. Antony Arokia Durai Raj, C. Rajendran, A genetic algorithm for solving the fixed-charge transportation model: two-stage problem, Comput. Oper. Res. 39 (9) (2012) 2016–2032.
[2] A. Balaji, N. Jawahar, A simulated annealing algorithm for a two-stage fixed charge distribution problem of a supply chain, Int. J. Oper. Res. 7 (2) (2010) 192–215.
[3] M.L. Balinski, Fixed-cost transportation problems, Nav. Res. Logist. Q. 8 (1) (1961) 41–54.
[4] C. Blum, Ant colony optimization: introduction and recent trends, Phys. Life Rev. 2 (4) (2005) 353–373.
[5] M.A. Contreras, W. Chung, G. Jones, Applying ant colony optimization metaheuristic to solve forest transportation planning problems with side constraints, Can. J. For. Res. 38 (11) (2008) 2896–2910.
[6] I. CPLEX, 11.0 users manual, ILOG SA, Gentilly, France, 2007.
[7] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.
[8] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolut. Comput. 1 (1) (1997) 53–66.
[9] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1) (1996) 29–41.
[10] D. Gaertner, K.L. Clark, On optimal parameters for ant colony optimization algorithms, in: IC-AI, Citeseer, 2005, pp. 83–89.
[11] W. Hackbusch, Multi-grid methods and applications, vol. 4, Springer-Verlag, Berlin, 1985.
[12] B. Hendrickson, R.W. Leland, A multi-level algorithm for partitioning graphs, SC 95 (1995) 28.
[13] W.M. Hirsch, G.B. Dantzig, The fixed charge problem, Nav. Res. Logist. Q. 15 (3) (1968) 413–424.
[14] F. Hutter, H.H. Hoos, K. Leyton-Brown, Automated configuration of mixed integer programming solvers, in: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer, 2010, pp. 186–202.
[15] F. Hutter, H.H. Hoos, K. Leyton-Brown, T. Stützle, Paramils: an automatic algorithm configuration framework, J. Artif. Intell. Res. 36 (1) (2009) 267–306.
[16] J.-B. Jo, Y. Li, M. Gen, Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm, Comput. Ind. Eng. 53 (2) (2007) 290–298.
[17] G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, Multilevel hypergraph partitioning: applications in vlsi domain, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 7 (1) (1999) 69–79.
[18] G. Karypis, V. Kumar, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, J. Parallel Distrib. Comput. 48 (1) (1998) 71–95.
[19] P. Korošec, J. Šilc, The multilevel ant stigmergy algorithm for numerical optimization, Facta Univ.-Ser.: Electron. Energ. 19 (2) (2006) 247–260.
[20] K. Kowalski, B. Lev, On step fixed-charge transportation problem, Omega 36 (5) (2008) 913–917.
[21] M. Leng, S. Yu, An effective multi-level algorithm based on ant colony optimization for bisecting graph, in: Advances in Knowledge Discovery and Data Mining, Springer, 2007, pp. 138–149.
[22] P. Lin, Constrained forest transportation planning problems, 2013, URL ⟨http://cs.uky.edu/~plin/ML_ACO1/⟩.
[23] P. Lin, M. Contreras, J. Zhang, W. Chung, Applying ant colony optimization to solve constrained forest transportation planning problems, in: Council on Forest Engineering Annual Meeting, Missoula, Montana, 2013.
[24] P. Lin, J. Zhang, M. Contreras, et al., Applying pareto ant colony optimization to solve bi-objective forest transportation planning problems, in: 2014 IEEE 15th International Conference on Information Reuse and Integration (IRI), IEEE, 2014, pp. 795–802.
[25] P. Lin, J. Zhang, M.A. Contreras, Automatically configuring aco using multilevel paramils to solve transportation planning problems with underlying weighted networks, Swarm Evolut. Comput. 20 (2015) 48–57.
[26] A. Noack, R. Rotta, Multi-level algorithms for modularity clustering, in: Experimental Algorithms, Springer, 2009, pp. 257–268.
[27] Y.H. Sheng, Research articles: special issues; fixed charge transportation problem and its uncertain programming model, IEMS 11 (2) (2012) 183–187.
[28] D.I. Steinberg, The fixed charge problem, Nav. Res. Logist. Q. 17 (2) (1970) 217–235.
[29] C. Walshaw, A multilevel approach to the travelling salesman problem, Oper. Res. 50 (5) (2002) 862–877.
[30] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Ann. Oper. Res. 131 (1–4) (2004) 325–372.