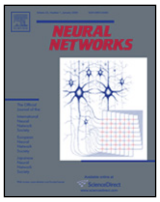
Neural Networks 24 (2011) 699–708

Contents lists available at ScienceDirect

Neural Networks

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



A deterministic annealing algorithm for the minimum concave cost network flow problem[☆]

Chuangyin Dang[a,*], Yabin Sun[a], Yuping Wang[b], Yang Yang[c]

[a] *Department of Manufacturing Engineering & Engineering Management, City University of Hong Kong, Hong Kong, China*
[b] *School of Computer Science and Technology, Xidian University, Xi'an, China*
[c] *School of Mathematics and Physics Science, Xuzhou Institute of Technology, Xuzhou, China*

ARTICLE INFO

Article history:
Received 3 August 2010
Received in revised form 1 March 2011
Accepted 16 March 2011

Keywords:
Concave cost
Network flow
Combinatorial optimization
Lagrange multiplier
Barrier function
Lagrange and barrier function
Descent direction
Iterative method
Deterministic annealing

ABSTRACT

The existing algorithms for the minimum concave cost network flow problems mainly focus on the single-source problems. To handle both the single-source and the multiple-source problem in the same way, especially the problems with dense arcs, a deterministic annealing algorithm is proposed in this paper. The algorithm is derived from an application of the Lagrange and Hopfield-type barrier function. It consists of two major steps: one is to find a feasible descent direction by updating Lagrange multipliers with a globally convergent iterative procedure, which forms the major contribution of this paper, and the other is to generate a point in the feasible descent direction, which always automatically satisfies lower and upper bound constraints on variables provided that the step size is a number between zero and one. The algorithm is applicable to both the single-source and the multiple-source capacitated problem and is especially effective and efficient for the problems with dense arcs. Numerical results on 48 test problems show that the algorithm is effective and efficient.

1. Introduction

The minimum concave cost network flow problem can be described as follows.

Let $G = (N, A)$ be a directed graph, where $N = \{1, 2, \dots, n\}$ is the set of n nodes of G and A is the set of arcs of G satisfying that $(i, j) \in A$ if and only if there is a directed arc from node i to node j. For any $(i, j) \in A$, let x_{ij} denote the flow from node i to node j, and assume that there is a maximum capacity $c_{ij} > 0$ on arc (i, j). Let $x = (x_{ij} : (i, j) \in A)^T$ and $f(x)$ denote the cost of the flow through the network. We assume that f is a continuously differentiable concave function. The minimum concave cost network flow problem is given as follows. Find a solution of

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & \sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = b_j, \quad j = 1, 2, \dots, n, \\ & 0 \le x_{ij} \le c_{ij}, \quad (i, j) \in A, \end{aligned} \tag{1}$$

where x_{ij} can be an integer variable or a continuous variable and b_j represents demand or supply of node j and satisfies that $\sum_{j=1}^{n} b_j = 0$. It is an NP-hard problem and has many applications including production planning, transportation and communication network design, and facility locations. Owing to the concavity of the objective function, an optimal solution occurs at a vertex of the polytope of the feasible region and local optimality does not imply global optimality, which means that classical approaches of nonlinear programming usually can only find a local minimum point. In order to solve the minimum concave cost network flow problem, a number of algorithms have been developed in the literature, which are referred to Fontes and Goncalves (2007), Fontes, Hadjiconstantinou, and Christofides (2006a, 2006b, 2003), Kim and Hooker (2002), Minoux (1986), Ortega and Welsey (2003), Pardalos and Rosen (1987), Trudeau (2009), etc. An excellent survey of algorithms for the minimum concave cost network flow problem can be found in Guisewite (1995) and the references therein. Most of these algorithms focus on the single-source uncapacitated problems or the problems for which the network is sparse, i.e., each node is connected to only a few nodes, say k nodes with $k \ll n$ (e.g., Fontes and Goncalves (2007), Fontes et al. (2003)). To solve the multiple-source and capacitated problems by the existing algorithms, one has to transform the problems into the single-source uncapacitated ones. However, this transformation process will introduce many

[☆] The authors are very grateful to the editor and reviewers for their valuable comments and suggestions. This work was partially supported by GRF: CityU 112809 of Hong Kong SAR Government.

[*] Corresponding author. Tel.: +852 27888429; fax: +852 27888423.
E-mail address: mecdang@cityu.edu.hk (C. Dang).

0893-6080/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2011.03.018

new nodes and arcs, which significantly increases the number of variables.

Generally speaking, if a minimum concave cost network flow problem has more nodes and arcs, the problem will be more difficult and time-consuming to solve. Also, for a fixed number of nodes n, when each node is only connected to a few nodes (say k nodes with $k \ll n$), that is, the network is sparse, the search for an optimal solution is much easier. With the increasing of k, the number of possible network flows increases exponentially, and the problem becomes more and more difficult to solve. Thus, for the problems with large k (we call the network is dense), the running time for the existing algorithms will dramatically increase. Especially, when $k = n - 1$ (the largest value of k), that is, each pair of nodes is connected by two opposite directed arcs, the problems in this case become much more difficult. These challenges appeal for more effective and efficient alternatives.

Neural Networks, since their emergence, have experienced significant advances in both theory and applications, especially in optimization, among which combinatorial optimization, due to Hopfield and Tank (1985), has become a popular topic in the literature of neural computation. Although initial results were disappointing, modified network dynamics and better problem mapping contribute significantly to solution quality (Gee, Aiyer, & Prager, 1993). Xu (1994) proposed a Lagrange and transformation method for combinatorial optimization neural net. Peterson and Soderberg (1989) mapped the graph partition problem onto a neural network with the graded neuron encoding, which can reduce the solution space by one dimension. Gee et al. (1993) presented a problem mapping evaluation method without recourse to purely experimental means. Gee and Prager (1994) proposed a rigorous mapping for quadratic 0–1 programming problems with linear equality and inequality constraints. After transforming variables with exponential functions, Urahama (1996) presented an analog solver for nonlinear programming problems with linear constraints. A feasible solution construction mechanism was introduced in Horio, Ikeguchi, and Aihara (2005) to improve the performance of the Hopfield-type chaotic neuro-computer system for quadratic assignment problems (QAPs). Bout and Miller (1990) and Wu (2004) applied the mean field annealing (MFA) algorithm for a solution of the graph bisection problem. Feig and Kuauthgamer (2006) and Ishii, Iwata, and Nagamochi (2007) proposed other algorithms for graph bisection problem. Waugh and Westervelt (1993) introduced a neural network architecture that is applicable in optimization. Wang and Xia (1998) designed a primal–dual neural network for assignment problems. Xia (2004, 2009) proposed projection neural networks for constrained optimization and variational inequalities. Xia and Feng (2007) designed a neural network for solving nonlinear projection equations. By combining deterministic annealing, self-amplification, algebraic transformations, clocked objectives, and softassign, an optimizing network architecture was constructed in Rangarajan, Gold, and Mjolsness (1996). Furthermore, special network models were constructed for the traveling salesman problem (Aiyer, Niranjan, & Fallside, 1990; Dang & Xu, 2001; Durbin & Willshaw, 1987; Wacholder, Han, & Mann, 1989; Wolfe, Parry, & MacMillan, 1994). Statistical mechanics as the underlying theory of optimization neural networks was studied in Simic (1990). A systematic investigation of such neural computational models for combinatorial optimization can be found in Berg (1996) and Cichocki and Unbehaunen (1993). Most of these algorithms are of deterministic annealing type, which is a heuristic continuation method that attempts to find the global minimum of the effective energy at a high temperature and track it as the temperature decreases. There is no guarantee that the minimum at a high temperature can always be tracked to the minimum at a low temperature, but the experimental results are encouraging (Yuille & Kosowsky, 1994).

In this paper, we adopt the idea of deterministic annealing to propose an algorithm for approximating a solution of the minimum concave cost network flow problem. The main idea of the algorithm is as follows. A Hopfield-type barrier function is used to deal with lower and upper bound constraints on variables, where the barrier parameter behaves as temperature in an annealing procedure and decreases to zero from a sufficiently large positive number which ensures that the barrier function is convex. Lagrange multipliers are introduced to handle linear equality constraints. The algorithm attempts to produce a high-quality solution by generating a minimum point of a barrier problem for a sequence of descending values of the barrier parameter. For any given value of the barrier parameter, in order to search for a minimum point of the barrier problem, the algorithm performs two major steps: one is to find a feasible descent direction by updating the Lagrange multipliers with a globally convergent iterative procedure, which forms the major contribution of this paper, and the other is to generate a point in the feasible descent direction, which has a nice feature that the lower and upper bounds on variables are always satisfied automatically provided that the step size is a number between zero and one. For any given positive value of the barrier parameter, we prove that the algorithm converges to a stationary point of the barrier problem. The algorithm solves both the single-source and the multiple-source capacitated problems in the same way and is especially effective and efficient for the dense network problems. Numerical results show that the algorithm is effective and efficient.

The rest of this paper is organized as follows. We introduce the barrier problem and derive several important theoretical results in Section 2. We describe the algorithm and show its convergence to a stationary point of the barrier problem for any given positive value of the barrier parameter in Section 3. We prove in Section 4 the global convergence of the iterative procedure for updating Lagrange multipliers to find a feasible descent direction. We present some numerical results in Section 5 to show that the algorithm is effective and efficient. Finally, we conclude the paper with some remarks in Section 6.

2. A Hopfield-type barrier function

We assume without loss of generality that $0 < b_j + \sum_{(j,k)\in A} c_{jk}$, $j = 1, 2, \ldots, n$, and that the problem has a strictly feasible flow. In order to solve (1), we introduce for each $(i, j) \in A$ a Hopfield-type barrier term,

$$h_{ij}(x_{ij}) = x_{ij} \ln x_{ij} + (c_{ij} - x_{ij}) \ln(c_{ij} - x_{ij}),$$

to incorporate $0 \le x_{ij} \le c_{ij}$ into the objective function, and obtain

$$\begin{aligned} \min \quad & e(x, \beta) = f(x) + \beta \sum_{(i,j)\in A} h_{ij}(x_{ij}) \\ \text{subject to} \quad & \sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = b_j, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{2}$$

where β is a barrier parameter. Instead of solving (1) directly, we consider a scheme that obtains a solution of (1) from a solution of (2) at the limit of $\beta \downarrow 0$.

Let $h(x) = \sum_{(i,j)\in A} h_{ij}(x_{ij})$. Then, $e(x, \beta) = f(x) + \beta h(x)$. Let

$$P = \left\{ x \,\middle|\, \begin{array}{l} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = b_j, \quad j = 1, 2, \ldots, n, \\ 0 \le x_{ij} \le c_{ij}, \quad (i, j) \in A \end{array} \right\}$$

and

$$B = \{x \mid 0 \le x_{ij} \le c_{ij},\ (i, j) \in A\}.$$