Systems Engineering — Theory & Practice
Volume 29, Issue 1, January 2009
Online English edition of the Chinese language journal

Cite this article as: SETP, 2009, 29(1): 111–117

RESEARCH PAPER

Heuristic Algorithm with Oscillation Strategy for a New Class of Assignment Problem

LUO Jia-xiang*[,1], TANG Li-xin[2], HU Yue-ming[1]

1. Engineering Research Center for Sophisticated Electronic Equipment Manufacture, Ministry of Education, South China University of Technology, Guangzhou 510640, China
2. The Logistics Institute, Northeastern University, Shenyang 110004, China

Abstract: A new class of assignment problem which roots in the optimization management of slabs in steel industry is considered in this article. Compared with the generalized assignment problem, flow constraints should be considered in this problem besides the capacity constraints when assigning items to knapsacks. This problem could be reduced to the generalized assignment problem, and so is NP-hard. A heuristic with oscillation strategy and long-term memory list is proposed to solve it. The oscillation strategy makes it possible that the local search oscillates between the feasible and infeasible solution spaces to find better feasible solutions. A long-term memory list is embedded to encourage the diverse moves of items, which improves the diversity of the algorithm. In order to testify the efficiency of the heuristic, 23 instances have been randomly generated for the computational experiments. The results show that for small-size instances, the maximum deviation of the heuristic from the optimal solution is 0.55% and for larger-size instances, the heuristic could find good solutions in a very short time.

Key Words: assignment problem; oscillation strategy; long-term memory list.

1 Introduction

A new class of assignment problem is considered in this article, which is an extended version of general assignment problem (GAP). This problem can be described as follows. The item j in set J (J={1, 2,..., n}) will be assigned to one of the knapsacks in set I={1, 2,..., m} to maximize the assigning profit, while the knapsack capacity constraints should be satisfied, as well as the flow constraint. The flow constraint indicates that the knapsacks belong to different types, and the total weight of items in each type should be no less than a predetermined value number.

This problem arises from the steel production. In most iron and steel companies, production is stimulated by customer orders, and the planning and scheduling of semi-products during the production is arranged according to the actual requirements of various orders. With the development of motor industry, household electrical appliance industry, and property industry, it is very usual that the customers order small amount but various steel products. However, the orders conflict the mass feature of steel production. For example, in steel-making stage, steel-making furnaces produce molten steel (it will be made to slabs in casting stage) in either 300 ton or 250 ton batches, which are not exactly the ordered amounts most of times. So the production of small amount and various products usually results in unsatisfied matching relationships between the slabs and orders. It is also the reason the produced slabs often exceed the actual required amount. Consequently, the customer satisfaction level decreases and production cost rises.

An effective way of avoiding these deficiencies is to reallocate the slabs for maximizing the matching profit by optimization method after releasing the original matching relationships. The new matching relationships should satisfy the following constraints: 1) the total slab weight allocated to each order should not exceed the order's requirement and 2) the slab weight allocated to the orders for each material flow should be not less than its requirement in order to ensure the balance of production, while the orders belong to different material flows (a material flow corresponds to one main downstream production operation for the products required by orders). If slabs are seen as items, orders as knapsacks and the material flow as knapsack's flow, the above problem is actually assigning items to knapsacks to maximize the assigning profit with the two constraints on knapsack capacity and knapsack flow. So the problem is an extended version of the GAP. However, compared with the GAP problem, the new problem is more complex for the new constraint on flow.

The GAP is an NP-hard problem[1], so it is claimed that the assigning problem with flow constraint is also an NP-hard problem. For such kind of problem, optimization method is hard to solve within a reasonable short time, and the intelligent optimization method is often adopted instead.

The discussed problem has two kinds of constraints, and the feasible space is relatively narrow. Oscillation strategy[2−3] is an effective search strategy for the problem with narrow feasible solution space. It comes from tabu

Received date: August 3, 2007
∗ **Corresponding author:** Tel: +86-20-8711-4489; E-mail: luojx@scut.edu.cn
Foundation item: Supported by the National Science Key Foundation of China (No.60835001), National Nature Science Youth Foundation of China(No.60804053), Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 200805611065)

search and makes the local search run alternatively between feasible solution space and infeasible solution space to obtain a better feasible solution. Literature [4] proposes a tabu search with oscillation strategy for 0–1 knapsack problem, which includes two stages: construction and destruction. In the construction stage, the search shifts from feasible space to infeasible space. In the destruction stage, the search shifts from infeasible space to feasible space. Literature [5] also uses the tabu search to solve 0–1 integer programming model and the search oscillates between feasible and infeasible solution space.

A heuristic algorithm with oscillation strategy is constructed for the problem with flow constraint in this article. The search alternates between feasible and infeasible solution space, furthermore, the algorithm has two features: 1) the algorithm parameters are dynamically adjusted and 2) a long-term memory list [1−2] is embedded to record the number of occurrences of an item in knapsacks in solutions during the search, and then the numbers are considered additional subobjective to encourage the diverse moves of items.

To evaluate the efficiency of the algorithm, simulation data is generated based on the practical production. The experimental results show that the algorithm can obtain satisfied solutions in a reasonable short time.

2 Problem model

The assigning problem with flow constraint has two features: 1) the weight of each item occupying the resources of different knapsacks is the same and 2) knapsacks are classified into different flows and the constraint on knapsack capacity should be satisfied, as well as the constraint on knapsack flow. The objective is to maximize assigning profit. The mathematical model for the problem is as follows.

Model parameters:

m - the total number of knapsacks;

n - the total number of items;

K - the total number of flows;

H_k - the knapsack subset belong to flow *k*;

p_{ij} - the profit of assigning item *j* to knapsack *i*;

a_j - the weight of item *j*;

b_i - the weight of knapsack i can be accepted (the capacity of knapsack *i*);

c_k - the minimum weight requirement for flow *k*.

Model variables:

$$x_{ij} = \begin{cases} 1, \text{ if item } j \text{ is assined to knapsack } i \\ 0, \text{ otherwise} \end{cases}$$

According the above parameters and variables, the model for the assigning problem with flow constraint is as follows:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_{j=1}^{n} a_j x_{ij} \le b_i, \;\; i = 1, 2, ..., m; \tag{2}$$

$$\sum_{i \in H_k} \sum_{j=1}^{n} a_j x_{ij} \ge c_k, \;\; k = 1, 2, ..., K; \tag{3}$$

$$\sum_{i=1}^{m} x_{ij} \le 1, \;\; j = 1, 2, ..., n; \tag{4}$$

$$x_{ij} = \{0, 1\}, i = 1, 2, ..., m; j = 1, 2, ..., n. \tag{5}$$

Formula (1) is the objective function, which is the total profit of assigning items to knapsacks. Constraint (2) requires that the knapsacks' capacity should not be exceeded. Constraint (3) indicates that the total weight of the items assigned to flow *k* should be not less than the required. Constraint (4) indicates that each item can only be assigned to one knapsack. Constraint (5) defines the ranges of the variables.

Omitting constraint (3), the model could be reduced to a typical GAP. In GAP, items are assigned to knapsacks and each in a condition that only one item can be assigned to one knapsack, while the constraints on knapsack capacity should be satisfied, and the objective is to maximize the total profit. The GAP is NP-hard[1], and so it claims that the assignment with flow constraint is also NP-hard. The model described above is an extension of the GAP.

Let z_{ik}=1, if knapsack *i* belongs to flow *k*, or else z_{ik}=0, and then constraint (3) can be transformed into the following constraints.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_j z_{ik} x_{ij} \ge c_k, \;\; k = 1, 2, ..., K \tag{6}$$

The transformed model is a linear integer programming, which could be solved into optimal by linear solver for small-scale size of instances. However, for the large-scale size of instances, it is very difficult for linear solver to obtain the optimal solution or even a satisfied feasible solution. Considering that the two kinds of constraints of the problem make the feasible solution space narrow, a heuristic with oscillation strategy is constructed.

3 Problem solving method

For the assigning problem with flow constraint, a heuristic with oscillation strategy is constructed. In the oscillation strategy, local search oscillates between feasible and infeasible solution spaces to search for better feasible solutions. Otherwise, long-term memory list is introduced to encourage the diverse moves of items to improve the global search ability.

The main idea of the algorithm is as follows: first, the initial solution is constructed according to the two constraints; second, the search with oscillation strategy makes an iterative search for a good solution in various neighborhoods. At each iteration, swap and insertion neighborhoods are searched in turn. For the neighbors of a solution, their quality is evaluated by the objective function value, the depth of infeasibility, and the frequencies of items assigned to knapsacks. If the search is in infeasible solution space for a predetermined number of continuous iterations, the oscillation strategy is adopted to guide the search come back to feasible solution space.