Proceedings of the 9th WSEAS International Conference on Applied Mathematics, Istanbul, Turkey, May 27-29, 2006 (pp263-271)

AN APPLICATION OF LAGRANGEAN DECOMPOSITION TO THE SCHEDULING OF HOT CHARGED ROLLING IN STEEL PRODUCTION

T.-C. Chen
Department of Information Management
National Formosa University
64, Wen-Hua Rd., , Huwei, Yulin, 632, TAIWAN

Abstract:-In this paper a Lagrangean decomposition technique for solving the scheduling problem for Hot Charged Rolling (HCR) in the continuous casting process which involves sequencing and grouping. This Lagrangian relaxation algorithm is proposed that incorporate two sets of constraints into the objective function after applying the variable splitting technique. The relaxed problem has a special structure and can be solved as three versions of subproblems. The subgradient algorithm is then used to maximize the Lagrangian dual. Two heuristic algorithms are also proposed to find good primal feasible solutions. Through the three versions of subproblems and two heuristic algorithms, the best one from six combinations is selected to incorporate with branch-and-bound algorithm to find the optimal solution when job size is large. Our methodology provides the optimal solutions for the difficult scheduling problem resulted from the continuous casting process.

Key-Words: Hot Charged Rolling, Lagrangian relaxation, Continuous casting.

1 Introduction

In the continuous casting process, the most modernized steel mills which have the capability of Hot Charged Rolling (HCR). The HCR is a modern steel manufacturing concept where the steel making, slab casting and hot coil production processes are physically integrated into one directly connected and synchronously controlled process.

The major advantage of the HCR is in the reduction of the slab inventory and the savings in energy cost normally wasted for reheating slabs for hot coil production. However, the scheduling task for the HCR becomes significantly more difficult than the one for the conventional steel manufacturing process.

The HCR process includes three processes: steel making, slab casting, and hot coil production. The melted pig iron produced at the blast furnace is converted into melted steel at the steel making process. The melted steel is produced in the converter having a certain capacity and shipped in an open ladle to the slab casting process. This melted steel in the open ladle is then put into the continuous caster, where the melted steel is formed into slabs. Then these slabs pass through the rollers at the hot strip mill to become hot coils.

The jobs to be scheduled for the HCR process are hot coils having the following attributes: weight, thickness, width, and quantity. The weights are within a certain range of tons. There are several different thickness groups and width groups. This scheduling problem for the HCR involves both grouping and sequencing.

For steel making, jobs must be grouped into lots called "charges". These charges are then grouped into larger lots called "casts". Charges in the same cast must be properly sequenced according to the charge sequencing rules, i.e., (1) Width must be nonincreasing; (2) Thickness must be nondecreasing; (3) Each charge will not weight more than the limitation.

The approach used in this paper is based on Lagrangian relaxation. This problem has a special structure that allows the Lagrangian problem to be decomposed into three easy problems: a minimum spanning tree, an assignment problem and a minimum cost network flow problem. The subgradient method is then used in a dual algorithm to tighten the Lagrangian lower bounds. We also have developed two heuristic methods to tighten the upper bounds because it may be that optimal values for the subproblems are not feasible. For this reason, it is worthwhile to seek a feasible solution by means of a heuristic. Our contribution is to present a dual-based heuristic procedure, incorporated in the Lagrangian algorithm, to generate good primal feasible solutions. The lower bound generated by the subgradient algorithm is then used to measure the quality of the heuristic solution in terms of the duality gap. A branch-and-bound algorithm based on Lagrangian relaxation algorithm is proposed to search for all optimal solutions.

2. Literature Survey

In 1970, Held and Karp [14] used a Lagrangian problem based on minimum spanning trees to devise a successful algorithm for the Traveling Salesman Problem. Because of their success, they were widely applied to scheduling problems, general integer programming problems, location problems, set covering problems, and so on.

Altinkemer and Gavish [2] present a 0-1 formulation of a vehicle delivery problem and give a number of heuristics for the problem. A lower bound, based upon Lagrangian relaxation and subgradient optimization, is used to evaluate the quality of the heuristic solutions. Admadi and Tang [1] present a paper dealing with the operation partitioning problem. This is the problem of assigning operations to machines so as to minimize the total movement of jobs between machines. They presented two 0-1 formulations of the problem and used Lagrangian relaxation, subgradient optimization and Lagrangian heuristics. Fetterolf and Anandalingam [7] use the Lagrangian relaxation technique to optimize the interconnection of local area networks. After two sets of constraints are relaxed, the problem is easily solved by decomposing it into two subsets of problems. The subgradient optimization and Lagrangian heuristic are applied. In this paper when the networks are large, it is hard to find feasible solutions just by the Lagrangian heuristic. Daskin and Panayotopoulos [4] solve the problem of assigning aircraft to routes to maximize profits in a hub and spoke network. The Lagrangian heuristic and the subgradient optimization are also used and two heuristics are developed for obtaining the feasibility when the Lagrangian solution is not feasible. Because of the heuristics, good feasible solutions are found more easily and the Lagrangian problem converged more quickly. Resnoso and Maculan [19] present a new Lagrangian decomposition scheme for integer linear programming problems. This scheme is based on a reformation of the problem by introducing one or more copies of some of the decision variables and a number of coupling constraints. By defining a Lagrangian relaxation of the copy and coupling constraints, it is possible to decompose the problem into two or more independent subproblems.

A branch-and-bound algorithm, using a lower bound obtained from the solution of linear assignment problem, is proposed by Rinnooy [21] for problems with a more general objective function. Some fathoming rules are presented by Etcheberry [5] and Fisher [11]. Potts [18] used a Lagrangian based branch-and-bound algorithm for the single machine sequencing where the heuristic method is used prior to the application of branch-and-bound. He applies the tree optimal heuristic of Morton and Dhara [16]. This method has the advantage that in some problems with only a few precedence constraints, the heuristic will indicate that the heuristic generated is optimal.
The paper applies the advantages of these papers. Like most of the papers the Lagrangian relaxation, subgradient optimization algorithm, and two Lagrangian heuristic methods proposed here to solve the Lagrangian problems. At each iteration of the Lagrangian heuristic we get a feasible solution from our heuristics so that the optimal solutions are more easily found. We also apply branch-and-bound to get the optimal solution for large job size problems.

3. Solution methology

The approach used in this paper is based on Lagrangin relaxation using variables splitting technique, this problem has a special structure that allows Lagrangian problem to be decomposed into three easy subproblems: Minimum spanning tree, assignment problem and minimum cost network flow problem. We called the combination of these three subproblems as version 1. In the version 2, the assignment problem of version has been transferred into generalized upper bound problem or multiple choice problems. In version 3, the constrain of X is a tree is replaced with the strong constraint that X is an "aborescence." Then relax as in version 2.

The other algorithm is developed as the branch and bound for big size problem. This algorithm still applies the technique of Lagrangian relaxation, which can be used at each node of the branching tree to obtain lower bounds and feasible solutions to avoid doing the unnecessary works. The results from this algorithm are time consuming and the number of branching nodes is huge when the size of the problem is large, though we can get the optimal solutions from this algorithm.

3.1 Model Formulation

A set of jobs is to be scheduled on a machine. For some pairs of jobs, no major setup is required if product immediately follows job , e.g., if the width and thickness of what:

- Sum of weight for each charge must be within a certain limit
- Width is nonincreasing and thickness is nondecreasing.

We wish to sequence the products so as to minimize the number of major setups required.

Represent the jobs by nodes in a network, with arc from node to node if node requires no major setup when if follows node. For example, Figure 1 represents a simple case in which the three quantities in parenthesis at each node represent the width, thickness and the weight, respectively, in the case in which we suppose the grades of the jobs are all the same. The nodes on a path through the network correspond to a sequence of jobs which can be processed with a single major setup. Any two such paths should be disjoint, i.e., should share no common jobs. So we should find the minimum number of disjoint paths which span all the nodes of a directed graph.

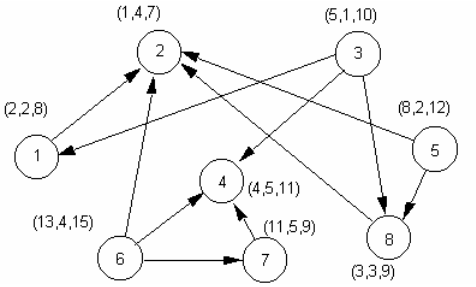

Fig.1 Example of network structure

In addition, we should impose the limitation of the total weight for each path, i.e., the sum of weight for each charge must be within a certain limit. Figure 2 represents one feasible solution with paths shown in bold if we suppose the weight limitation is 35 units of weight limit.

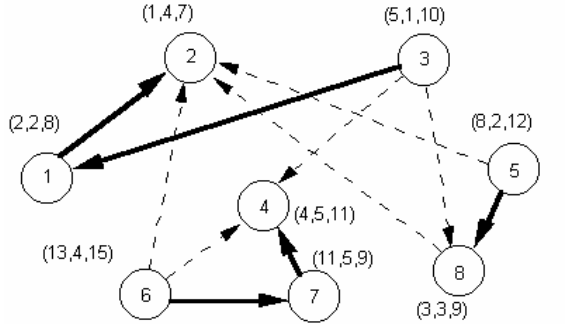

Fig.2 A feasible solution

3.1.1 Mathematical Programming Model

Given the directed graph (digraph) G:
G=(N,A), where N={1,2,3,.....n}= set of nodes.
A= set of arcs ( ),
we define the variables:

$$X_{ij} = \begin{cases} 1 & \text{if arc (i, j) is included on a path} \\ 0 & \text{otherise} \end{cases}$$

f_{ij} = Flows of arc (i, j)

W = Weight limit

w_i = Weight of node i

Clearly
X_{ij} = 1 for at most one j for each i
That is, at most one arc enters node j and at most one arc leaves node i. Thus, we have the constraints as follows:

$$\sum_{j=1}^{n} X_{ij} \le 1 \qquad \text{for each } i \in N$$

However, the above constraints permit circuits. For this reason we have to add a constraint that the edges of the subgraph indicated by X form a "forest", i.e., a collection of trees. Each tree is a subgraph containing no cycle. In order to facilitate defining the objective function (which is to be the number of paths) in terms of X, define a new node 0. Let G'=(N',A') where

$$N = N \cup \{0\}$$
$$A = A \cup \{(0,1),(0,2),.....(0,n)\}$$

Let $X_{0j} = \begin{cases} 1 & \text{if node i is the beginning of a path} \\ 0 & \text{otherwise} \end{cases}$

When we add the node 0 to the network, then the network would appear as Figure 3.

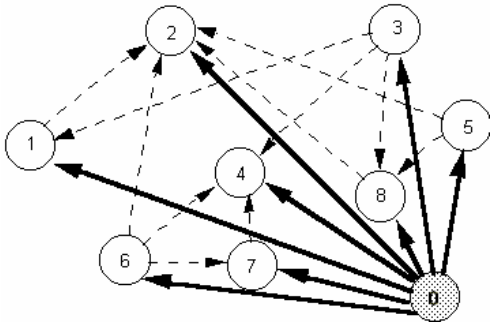

Fig.3 A redefined network

After adding the constraint to form a new problem and solving it, we find that the optimal solution of this new problem is a spanning tree. See Figure 4. This property will make it more convenient to handle this problem.

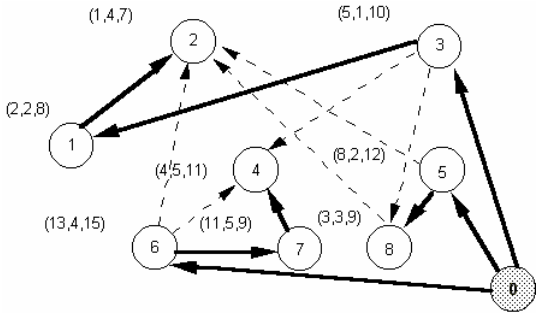

Fig.4. A collection of trees

3.1.2 Formulation

Problem (D)

Z = Minimize $\sum_{j=1}^{n} X_{0j}$ (1)

subject to

$X \in T$ = set of all spanning tree of G' (2)

$\sum_{j=1}^{n} X_{ij} \le 1$ for each $i \in N$ (3)

$\sum_{i=0}^{n} X_{ij} = 1$ for each $j \in N$ (4)

$x \in \{0,1\}$ for each $(i, j) \in A$ (5)

$\sum_{j=1}^{n} f_{0j} = \sum_{j=1}^{n} w_j$ (6)

$\sum_{k} f_{ki} = w_i + \sum_{j} f_{ij}$ for each $i \in N$ (7)

$f_{ij} \le W X_{ij}$ for each $i \in N$ and $j \in N$ (8)

$f_{ij} \ge 0$ for each $(i, j) \in A$ (9)

The constraint (6) specifies that the total weight of the flow entering each path should be equal to the sum of all job's weights. The constraint (7) is the conservation of flow in the network, where the w_i is the net flow into node i, i.e., the weight of job i. The constraint (8) indicates that if arc (*i, j*) is not included on the spanning tree, then there is no flow in that arc.

This formulation appears to be a good candidate for Lagrangian relaxation because its structure can be separated into three parts, the minimum spanning tree problem, the assignment problem and the minimum cost network flow problem.

As we know the spanning tree constraint is not easily expressed a system of explicit linear constraints, so that relaxing them is problematic. We use the variable "splitting" method to solve this difficulty.

3.1.3 Variable "Splitting"

For each variable X_{ij} of the problem, define a variable Y_{ij}. If we require that X be a spanning tree and Y be a feasible assignment, then we have to add a new set of constraints to our problem, namely $X_{ij} = Y_{ij}$ for each *i* and *j*, $(i, j) \in A$. The problem is then written as follows:

Z = Minimize (10)

subject to

$X \in T$ = set of all spanning tree of G' (11)

$\sum_{j=1}^{n} Y_{ij} \le 1$ for each $i \in N$ (12)

$\sum_{i=0}^{n} Y_{ij} = 1$ for each $j \in N$ (13)

$\sum_{j=1}^{n} f_{0j} = \sum_{j=1}^{n} w_j$ (14)

$\sum_{k} f_{ki} = w_i + \sum_{j} f_{ij}$ for each $i \in N$ (15)

$f_{ij} \le W Y_{ij}$ for each $i \in N$ and $j \in N$ (16)

$X_{ij} = Y_{ij}$ for each $i \in N$ and $j \in N$ (17)

$Y \in \{0,1\}$ for each $(i, j) \in A$ (18)

$f_{ij} \ge 0$ for each $(i, j) \in A$ (19)

for some specified weight which distributes the cost between the two sets of variables $(0 \le \alpha \le 1)$.

3.2 Lagrangian Relaxation

In the problem D, the constraints (16) and (17) are problematic, because the constraint (16) couples the binary decision variables of assignment problem with the network flow variables and the constraint (17) couples the minimum spanning tree variables with the assignment problem variables. If they are relaxed and incorporated into the objective function with Lagrange multipliers, the resulting Lagrangian problem is as follows:

Problem (D1)

$$\Phi(\lambda, \mu) = Min.\ \alpha \sum_{j=1}^{n} X_{0j} + (1-\alpha) \sum_{j=1}^{n} Y_{0j} + \sum_{i=0}^{n} \sum_{j=1}^{n} \lambda_{ij} (X_{ij} - Y_{ij}) + \sum_{i=0}^{n} \sum_{j=1}^{n} u_{ij} (f_{ij} - W Y_{ij})$$

subject to

$X \in T$ = set of all spanning tree of G'

$\sum_{j=1}^{n} Y_{ij} \le 1$ for each $i \in N$

$\sum_{i=0}^{n} Y_{ij} = 1$ for each $j \in N$

$\sum_{j=1}^{n} f_{0j} = \sum_{j=1}^{n} w_j$

$\sum_{k} f_{ki} = w_i + \sum_{j} f_{ij}$ for each $i \in N$

$Y \in \{0,1\}$ for each $(i, j) \in A$

$f_{ij} \ge 0$ for each $(i, j) \in A$

where the λ_{ij} is the Lagrangian multiplier of relaxed constraint (17), and $\mu_{ij} \ge 0$ is the Lagrangian multiplier of relaxed constraint (16).

After these two constraints are relaxed, the constraints of problem (D1) are separable in the three sets of variables X, Y, and *f*. This allows us to decompose the problem into three subproblems

which are designated by the coefficients of the three sets of variables. We state this subproblem as follows:

Problem (D1-1): Minimum spanning tree

$$\Phi_X(\lambda) = Min. \sum_{j=1}^{n} (\alpha + \lambda_{0j}) X_{0j} + \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_{ij} X_{ij}$$

subject to

$X \in T$ = set of all spanning tree of G'

Problem (D1-2): Assignment problem

$$\Phi_Y(\lambda, \mu) = Min. \sum_{j=1}^{n} (1 - \alpha - \lambda_{0j} - \mu_{0j} W) Y_{0j} - \sum_{i=1}^{n}\sum_{j=1}^{n} (\lambda_{ij} + \mu_{ij} W) Y_{ij}$$

subject to

$\sum_{j=1}^{n} Y_{ij} \le 1$ for each $i \in N$

$\sum_{i=0}^{n} Y_{ij} = 1$ for each $j \in N$

$Y \in \{0,1\}$ for each $(i, j) \in$ A

Problem (D1-3): Uncapacitated minimum cost network flow problem

$$\Phi_F(\mu) = Min. \sum_{i=0}^{n}\sum_{j=1}^{n} \mu_{ij} f_{ij}$$

subject to

$$\sum_{j=1}^{n} f_{0j} = \sum_{j=1}^{n} w_j$$

$$\sum_{k} f_{ki} = w_i + \sum_{j} f_{ij} \quad \text{for each } i \in N$$

$$f_{ij} \ge 0 \quad \text{for each } (i, j) \in A$$

In this problem, node 0 serves as a source for the flows, and each node $i \in N$ has a demand for w_i units. Because of the lack of capacities on the arcs, the minimum cost flow node 0 to node *i* is along the shortest path from node 0 to node *i*.

For any matrix λ_{ij} and μ_{ij} of the Lagrangian multipliers $(\mu_{ij} \ge 0)$, the sum of the optimal values of the three subproblems provides a lower bound on the optimal value of the original problem. $\Phi(\lambda, \mu) = \Phi_X(\lambda) + \Phi_Y(\lambda, \mu) + \Phi_F(\mu) \le Z^*$, where Z^* is the optimal solution of the original problem Z. It may be that the optimal values of X and Y for the subproblems are never feasible paths. For this reason, it is worthwhile to seek a feasible solution (which also provides an upper bound) by means of heuristic algorithms which use information provided by the Lagrangian multipliers. This feasible solution constitutes an upper bound on the optimal solution to our problem. That will be described in Section 3.4.

3.3 The Subgradient Optimization

To tighten the bound generated by the Lagrangian relaxation, it is desirable to find the vectors λ and μ such that the objective $\Phi(\lambda, \mu)$ is maximized. That is, find the optimal dual value Z_d such that

$Z_d = Max\Phi(\lambda, \mu)$ subject to $\mu \ge 0$

It is well known that the function $\Phi(\lambda, \mu)$ is concave and differentiable at all points, except where the Lagrangian relaxation has multiple optimal solutions (see Fisher [7]). Unfortunately, these points occur frequently, and thus unconstrained optimization using gradients is not acceptable. A method that has gained wide acceptance over the past several years is subgradient optimization. The application of this method to integer programming was pioneered by Held and Karp [14], and by Held, Wolfe and Crowder [15]. At differentiable (λ, μ) points the subgradient set of consists of a singleton equal to the gradient, while at nondifferentiable points the subgradient set consists of all of the gradients of all underestimating linear function which are equal to Φ at (λ, μ). Like gradients, the subgradients have the property that they point in a direction of ascent.

The search for the optimal dual variables and can be performed by subgradient optimization. One of the components of the subgradient of the dual objective, $\Phi(\lambda, \mu)$, is the matrix

$\Delta_\lambda = (\delta_{ij}^{\lambda})$ where $\delta_{ij}^{\lambda} = (X_{ij} - Y_{ij})$

The other component of the subgradient is as that:

$\Delta_\mu = (\delta_{ij}^{\mu})$ where $\delta_{ij}^{\mu} = (f_{ij} - WY_{ij})$

The adjustments of the two multiplier vectors are as follows:

$$\lambda_{new} = \lambda_{old} + \tau \frac{(Z^* - \Phi(\lambda_{old}, \mu_{old}))}{\|\Delta_\lambda\|^2} \Delta_\lambda$$

$$\mu_{new} = \mu_{old} + \tau \frac{(Z^* - \Phi(\lambda_{old}, \mu_{old}))}{\|\Delta_\mu\|^2} \Delta_\mu$$

where τ is a step size parameter satisfying $0 < \tau \le 2$, and Z^* is an upper bound on $\Phi(\lambda, \mu)$, i.e., the value of the best known primal feasible solution, frequently obtained by applying a heuristic algorithm as stated in the next section to the original problem (D). The denominator is the sum of the squares of the elements in the subgradient.

3.4 The Heuristic Algorithms

If the solution of the Lagrangian relaxation satisfies all of the relaxed constraints, the solution is feasible for the original problem. In this case, we can then use the value of the original objective function to update the upper bound. In our experience, however this

rarely happens. This is not surprising because of the very large number of relaxed constraints, all of which must be satisfied for the Lagrangian solution to be primal feasible.

In other words, it is possible that the optimal values of X and Y for the subproblems are never feasible paths in most cases. For this reason, to deal with the likely infeasibility of the solution to the relaxed problem, two heuristic methods have been developed to convert an infeasible Lagrangian solution into a feasible primal solution. They are the Greedy algorithm and the Random-search algorithm and are described as follows:

1. The greedy algorithm:

Initially, the path set P and Q are empty $(\mathrm{P} \leftarrow Q \leftarrow \varnothing)$.

(a) If all nodes lie on a path, i.e., $N = Q$, Stop. Else, begin a new path by selecting the node $i^* \in N - Q$,which minimizes λ_{0i}

Let $\mathrm{P} \leftarrow \mathrm{P} \cup \{(0, i^*)\}$ and $Q \leftarrow \{i^*\}$, $Q = Q \cup Q$,

(b) If {(i*, j) : j does not lie on a path, i.e., $j \in N - Q - Q$ } is empty, go to step (a). Otherwise, let. $j^* \leftarrow \text{argmin}\{\lambda_{i^* j} : j \in N - Q - Q\}$

(c) If $\sum w_i \geq W, i \in Q \cup \{j^*\}$, go to step (a). Otherwise go to (d).

(d) Let $\mathrm{P} \leftarrow \mathrm{P} \cup \{(\mathrm{i}, i^*)\}$, $Q \leftarrow Q \cup \{j^*\}$, and $i^* \leftarrow j^*$; Return to step (b).

2. The random search algorithm:

(a) If all nodes lie on a path, i.e., N=Q', Stop. Else, begin a new path by selecting the node $i^* \in N - Q$ which minimizes λ_{0i}

Let $\mathrm{P} \leftarrow \mathrm{P} \cup \{(0, i^*)\}$ and $Q \leftarrow \{i^*\}$

(b) If { (i^*, j) $\in A$: j does not lie on a path, i.e., $j \in N - Q - Q$ } is empty, go to step (a). Otherwise, the choice of j^* is random, with probability depending upon the current value of the Lagrange multipliers λ^*_{ij}. Specially, randomly select j^* from the set $N - Q - Q$ where the probability of selecting j^* is stated as follows: Fixed: i , we define

$C_{\max} = \max\{\lambda_{ij} + \mu_{ij} : (i, j) \in A\}$

$C_{\min} = \min\{\lambda_{ij} + \mu_{ij} : (i, j) \in A\}$

$P_i = C_{\max} + \theta(C_{\max} - C_{\min}) - (\lambda_{ij} + \mu_{ij})$, where θ=0.25.

$P_i = \dfrac{P_i}{\sum P_i}$

(c) If $\sum w_i \geq W, i \in Q \cup \{j^*\}$, go to step (a). Otherwise, go to (d).

(d) Let $\mathrm{P} \leftarrow \mathrm{P} \cup \{(\mathrm{i}, i^*)\}$, $Q \leftarrow Q \cup \{j^*\}$, and $i^* \leftarrow j^*$; Return to step (b).

3.5 Other Possible Relaxations

Other relaxations are possible. In Section 3.3, our relaxation decomposed into three subproblems, namely minimum spanning tree problem, assignment problem and the minimum cost network flow problem. We will refer to this relaxation as version 1. The other two possible relaxation we describe the Versions 2 and 3 are as follows:

Version 2: Relax, in addition to those relaxed in the approach just presented, the constraint on the in-degree of each node from the original problem (10):

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each j} \in \text{N}$$

The subproblem in Y is then a simple GUB (generalized upper bound) problem, or "multiple choice" problem, more easily solved than the assignment problem

The objective function of version 2 is

$$Min.\ \alpha \sum_{j=1}^{n} X_{0j} + (1-\alpha) \sum_{j=1}^{n} Y_{0j} + \sum_{i=0}^{n} \sum_{j=1}^{n} \lambda_{ij} (X_{ij} - Y_{ij}) + \sum_{j=1}^{n} \nu_j \left(\sum_{i=0}^{n} Y_{ij} - 1 \right) + \sum_{i=0}^{n} \sum_{j=1}^{n} \mu_{ij} (f_{ij} - W Y_{ij})$$

Subject to constraints (11), (12), (14), (15), (16), (18) and (19).

where υ is a dual Lagrangian multiplier vector for the constraint (13).

This vector ν is adjusted by

$\nu_{new} = \nu_{old} + \tau \dfrac{(Z^* - \Phi(\nu_{old}))}{\|\Delta_\nu\|^2} \Delta_\nu$ where $\Delta_\nu = (\delta^\nu_{ij})$ and $\delta^\nu_{ij} = (Y_{ij} - 1)$

This problem still could be decomposed into three subproblems, namely:

1. Minimum Spanning Tree subproblem

$$\Phi_X(\lambda) = Min. \sum_{j=1}^{n} (\alpha + \lambda_{0j}) X_{0j} + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij} X_{ij}$$

subject to constraint (11)

2. Generalized Upper Bound subproblem

$$\Phi_Y(\lambda, \mu) = Min. \sum_{j=1}^{n} (1 - \alpha - \lambda_{0j} + \nu_j - \mu_{0j} W) Y_{0j} + \sum_{i=1}^{n} \sum_{j=1}^{n} (\nu_j - \lambda_{ij} - \mu_{ij} W) Y_{ij} - \sum_{j=1}^{n} \nu_j$$

subject to constraints (12) and (18)

3. Minimum Cost Networks Flow subproblem

$$\Phi_F(\mu) = Min. \sum_{i=0}^{n} \sum_{j=1}^{n} u_{ij} f_{ij}$$

subject to constraints (14), (15) and (16).

Version 3: In this version, the same constraints are relaxed as in version 2, the constraint that X is a tree

is replaced by the stronger constraint that X is an "arborescence". An arborescence can be thought of as a directed tree that can be used as a grapevine. The root of an arborescence is the unique vertex included in the arborescence that has no arcs directed into it. A branching is defined as a forest in which each tree is an arborescence. The maximum branching algorithm can be used to find a maximum spanning arborescence rooted at a specified vertex, say vertex 0.

3.6 Branch-and-Bound Algorithm

In case the upper bound (provided by the heuristics) and the lower bound (provided by Langrangian relaxation) do not converge within a specified number of iterations, a branch-and-bound algorithm will be used to solve this problem. In this section we give a complete description of the branch-and-bound algorithm, except for the lower bounding rule which has already been described in detail in Section 3.3. In particular, a heuristic method is still used to find the feasible solution, i.e., upper bounds. The branching rule and the search strategy are described as follows.

The branching rule in the algorithm consists of applying the Lagrangian relaxation algorithm to a particular edge. We identify the set of edges which are incident to the nodes with excess in-degree or out-degree, and then the particular edge with maximum length is selected from this set of edges. We define the multipliers of spanning tree as the distance for the set of edges. After selecting the particular edge, we first exclude it from the network to generate a new node of the searching tree and then apply the Lagrangian relaxation algorithm to check if the subproblem should be fathomed or not. If it is not fathomed by excluding the edge then we select the next edge for branching until the subproblem is fathomed. After a subproblem fathomed by excluding an edge, the edge is then forced into the network to form another subproblem, i.e., node of the branching tree. If any of the following possibilities occurs, the subproblem should be fathomed, i.e., no further branching is done.

(1) The greatest lower bound (or its ceiling) computed for the subproblem is greater than or equal to the current incumbent.
(2) The greatest lower bound at the node is less than the current incumbent but a feasible solution can not be obtained when any edge is selected from the set of candidates for branching.

When all subproblems are fathomed, the solution corresponding to the current value of is optimal. The branch-and-bound algorithm using Lagrangian relaxation is described in the flow chart in Figure 5.

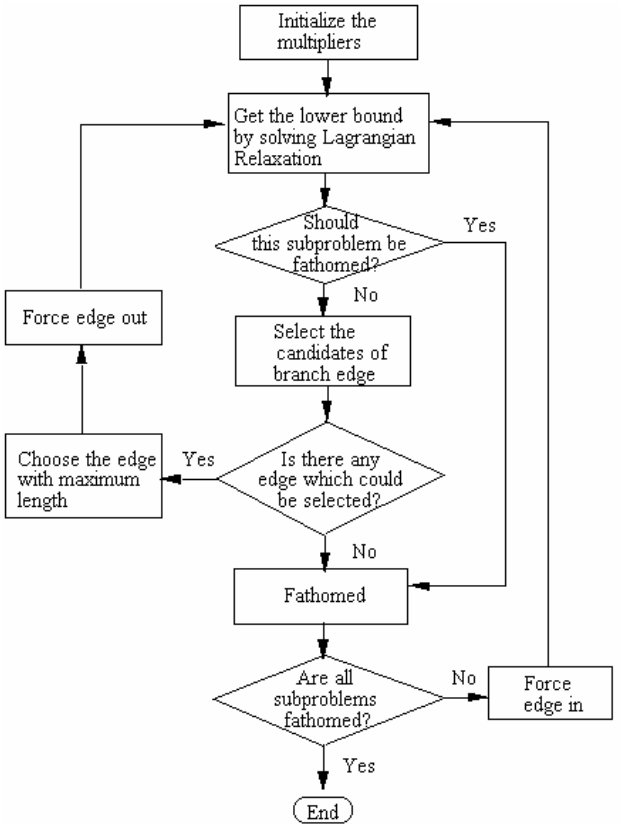

Fig.5 Branch-and-Bound algorithm

4. Results and Discussion

In this section the strengths and weaknesses of the combinations of three versions of Lagrangian relaxation with two heuristic methods will be investigated to find which combination is the most effective. And then the best combination will be utilized in the branch-and-bound method to get good bounds.

The Lagrangian algorithm was tested using several problems which were generated randomly. The respective problem sizes, the total iterations and CPU time on a HP-UX series 700 workstation are described in Table 1.

The duality gap is characterized by ZSTAR, the primal optimum obtained by enumeration, and GLB, the optimal dual obtained with Lagrangian relaxation. The duality gap is defined as follows:

$$\text{Duality gap} = \frac{(\text{ZSTAR} - \text{GLB})}{\text{ZSTAR}} \times 100\%$$

Now the six combinations of three relaxations with two heuristic methods will be studied, with the aim of getting the best one from these combinations. Five sets of results obtained by these six combinations are shown from Table 1.

For each six different size jobs, thirty randomly generated problems were tested here. Each problem

was run 35 times with different initial values for the multipliers for each of the six combinations, where we set the weight limit large enough that convergence is not hard to achieve. Then mean iteration counts and CPU time were then computed.

Table 1. Results from six test problems with 35 runs for each problems

LRP	Heuristic \ # of jobs		8 jobs	10 jobs	15 jobs	20 jobs	30 jobs	50 jobs
Version 1	Greedy	iter	3.615	5.167	14.758	20.737	32.111	T
		cpu	2.397	7.998	51.995	217.585	1143.81	T
	Random	iter	4.036	5.704	14.074	19.559	31.23	T
		cpu	2.687	7.851	50.405	191.923	1162.28	T
Version 2	Greedy	iter	6.67	7.767	18.633	16.3	26.286	60.103
		cpu	0.963	1.378	5.695	7.22	21.436	118.371
	Random	iter	10.233	18.076	54.767	91.429	F	F
		cpu	1.478	3.243	16.001	37.37	F	F
Version 3	Greedy	iter	4.1	8.867	46.36	81.8	F	F
		cpu	1.038	2.791	24.105	61.227	F	F
	Random	iter	3.373	8.367	49.5	65.167	F	F
		cpu	0.887	2.525	26.221	50.669	F	F

Note: F indicates that the Lagrangian relaxation did not converge after 250 iterations; T indicates that the Lagrangian relaxation did not converge within 1200 seconds CPU time.
# of jobs : number of jobs of this problem
iter : The mean value of the total iterations.
cpu : The mean value of the total CPU time.
LRP : Lagrangian Relaxation Problem.

According to the results in the table, the combination of version 2 with the greedy heuristic is the best one of these six combinations, because as the number of job increases, the mean iterations and the mean CPU time increase smoothly. Compared with the version 1 and version 3, the combination of version 2 and greedy heuristic can accomplish convergence and take the least time when the number of jobs is 50. From this observation, the combination of version 2 with the greedy heuristic is more attractive than other five.

It seems that the combination of the version 2 with the greedy heuristic is the best one to search for the optimal solution so that we will incorporate this combination into the branch-and-bound method for getting the optimal solution. When the problem is such that convergence is not achieved, we need to utilize the branch-and-bound method to seek for the optimal solution.

Six different sizes of jobs were test by the branch-and-bound algorithm to find the optimal solutions. For each size of problem includes thirty randomly generated problems. At each branching node a maximum of 150 iterations of the Lagrangian relaxation will be performed if it cannot be fathomed. Then an edge will be selected to form the next pair of branching nodes. The average of the cpu time of finding optimal solutions for the large size of jobs is described in Figure 6.

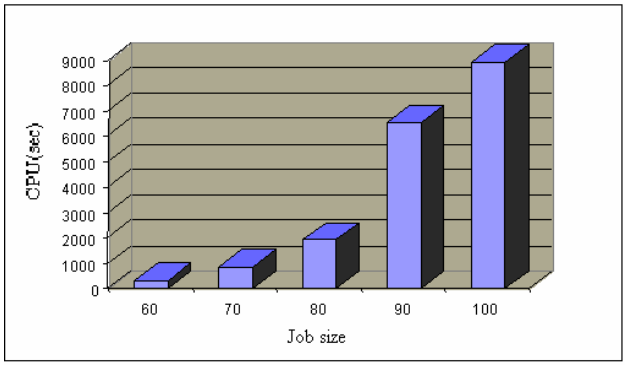

Fig.6 The cpu time from test problems by branch-and-bound algorithm.

As expected, CPU time increases when the number of jobs is increased. But when the jobs are increased form 80 to 90, the CPU time will suddenly more than double. When there are 100 jobs, in general, they will take much more time, the CPU time increases exponentially with the number of jobs for most cases.

After testing the six combinations of three Lagrangian relaxation with two heuristic methods, we have concluded that version 2 and the greedy heuristic is the best combination to solve this Lagrangian problem, and that means this combination is good enough to be applied in the branch-and-bound method for finding the optimal solution when the job size is more than fifty.

While the problem size is less than 50, the optimal solution can be obtained by using Lagrangian heuristic. However, if the problem is more than 60 jobs, this combination of version 2 and greedy heuristic fails to converge (gap>5%). It indicates that while job number is greater than 60, an alternative solution method should be developed for finding the optimal solutions. The obvious advantage of this Lagrangian heuristic is that it takes less CPU time for solving the problem than branch-and-bound did, especially in the large size problems.

5. Conclusions

This research studies job scheduling involving sequencing and grouping problems encountered by the continuous casting process in which the number of the major setup plays a big role in the production cost. When applying the variable splitting technique and relaxing two sets of constraints into the objective

function with Lagrange multipliers, the problem can be separated into three subproblems. Based on the three subproblems with more relaxation, the other two versions of subproblems were constructed.

Three versions of ubproblems were proposed to provide the lower bound and two heuristic methods were designed to convert the Lagrangian solution into a primal feasible solution. Tests comparing the six combinations of three relaxation versions with two heuristics indicate that the version 2 with the greedy heuristic is the most effective at solving the problem to guarantee get the optimal solutions or 'near' optimal solution when the number of the jobs is more than 60. In practice, the problem size usually falls into the small to moderate categories. The hope is that this research can be considered as a prototype or demonstration preparatory to further research.

References:

[1] Ahmadi, R. H. and Tang, C.S., An operation partitioning problem for automated assembly system design, *Operations Research*, Vol.39, 1991, pp.824-835.

[2] Altinkemer, K. and Gavish, B., Parallel savings-based heuristics for the delivery problem. *Operations Research*, Vol.39, 1991, pp.454-469.

[3] Bricker, D. L., Class notes of Integer programming and network flows, University of Iowa, Iowa City, Iowa, 1993.

[4] Daskin, Mark S. and Nicholaos D. Panayotopoulous. A Lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks, *Transportation Science*, Vol.23, No.2, 1989, pp.91-99.

[5] Etcheberry, J., The set-covering problem: A new implicit enumeration algorithm. *Operations Research*, Vol.25, No.5, 1977, pp.760-772.

[6] Evans J.R. and Minieka E., *Optimization Algorithms for Networks and Graphs*, Second Edition Revised and Expended, Marcel Dekker, Inc.,1992.

[7] Fetterolf P. C. and Anandalingam G.. A Lagrangian relaxation technique for optimizing interconnection of local area network, *Operations Research*, Vol.40, No. 4, 1992, pp.678-688.

[8] Fisher, Marshall L., The Lagrangian relaxation method for solving integer programming problems, *Management Science*, Vol.27, No.1, 1981, pp.1-17.

[9] Fisher, Marshall L, An applications oriented guide to Lagrangian relaxation, *Interfaces* Vol.15, No.2, 1985, pp.10-21.

[10] Fisher, Marshall L., Optimal solution of scheduling problems using Lagrange multipliers: Part I, *Operations Research*, Vol.21, 1973, pp.1114-1127.

[11] Fisher, Marshall L., Jaikumar, R. and Wassenhove, Luk N. Van., A multiplier adjustment method for the generalized assignment problem, *Management Science*, Vol.32, No 9, 1986, pp.1095-1103.

[12] Gavish, B., Topological design of centralized computer networks-formulations and algorithms, *Networks*, Vol.12, 1982, pp.355-377.

[13] Geoffrion, A. M., Lagrangian relaxation for integer programming, *Mathematical Programming Study*, Vol.2, 1974, pp.82-114.

[14] Held, M. and Karp, R. M., The traveling-salesman problem and minimum spanning trees, *Operations Research*, Vol.18, 1970, pp.1138-1162.

[15] 15. Held, M., Wolfe, P. and Crowder, H. D., Validation of subgradient optimization, *Mathematical Programming*, Vol.6, No.1, 1974, pp.62-88.

[16] 16. Morton, T. E. and Dharan, B. G., Algorithms for single-machine sequencing with precedence constraints, *Management Science*, Vol.24, 1978, pp.1011-1020.

[17] Peter B. L and Debra J. Hoitome., Scheduling of manufacturing systems using the Lagrangian relaxation technique, *IEEE Transactions: Automatic control*, Vol.38, No.7, 1993, pp.1066-1079.

[18] Potts, C N., A Lagrangian based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time, *Management Science*, Vol.31, 1985, pp.1300-1311.

[19] 19. Reinoso, H., and Maculan, N., Lagrangean decomposition in integer linear programming: A new scheme, *Information System and Operation Research*, Vol.30, No.1, 1992, pp.1-5.

[20] 20. Reeves, C. R., *Modern heuristic techniques for combinatorial problems*, New York, John Wiley & Sons, Inc, 1993.

[21] 21. Rinnooy Kan. A.H.G., Lageweg. B. J. and Lenstra, J.K., Minimizing total costs in one-machine scheduling, *Operation Transaction*, Vol.23, 1975, pp.908-927.