Available online at www.sciencedirect.com
SCIENCE DIRECT
Computers and Chemical Engineering 30 (2006) 778–789
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Lagrangean decomposition using an improved Nelder–Mead approach for Lagrangean multiplier update

Dan Wu, Marianthi Ierapetritou *

Chemical and Biochemical Engineering Department, Rutgers University, 98 Brett Road, Piscataway, NJ 08855-0909, United States

Received 5 July 2005; received in revised form 22 November 2005; accepted 5 December 2005
Available online 7 February 2006

Abstract

Lagrangean decomposition has been recognized as a promising approach for solving large-scale optimization problems. However, Lagrangean decomposition is critically dependent on the method of updating the Lagrangean multipliers used to decompose the original model. This paper presents a Lagrangean decomposition approach based on Nelder–Mead optimization algorithm to update the Lagrangean multipliers. The main advantage of the proposed approach is that it results in improved objective function values for the majority of iterations. The efficiency of the proposed approach is illustrated with examples from the literature and the solution of scheduling problems.

Keywords: Lagrangean decomposition; Nelder–Mead algorithm; Scheduling; Uncertainty

1. Introduction

Mathematical programming techniques have increasingly played an important role in chemical process industry. Optimization models are widely developed in all areas of design and operation in order to achieve maximum production with a minimum cost. The complexities of reality however require the use of large-scale models that often prevent the convergence to the optimal solution. Therefore, a number of publications have been focused on decomposition techniques that reduce the problem size and achieve solution in affordable computational time (Benders, 1962; Dantzig, 1963; Fisher, 1981). Lagrangean relaxation was original developed by Held and Karp (1970) and successfully applied to many optimization problems such as production scheduling (Fisher, 1973), planning (Graves, 1982; Gupta & Maranas, 1999) and lot-sizing problems (Diaby, Bahl, Karwan, & Zionts, 1992; Thizy & Wassenhove, 1985).

The idea of Lagrangean relaxation is based on the existence of "hard" constraints in the mathematic description of the optimization problems that increase the computational complexity of the solution approach. As shown in Fig. 1, assuming that the first set of constraints are the complicating ("hard") constraints, Lagrangean relaxation proceeds by relaxing these constraints and penalizing the constraint violation in the objective function thus reducing the computational complexity of the solution procedure since the problem with the remaining constraints is easier to solve.

Since the set of Lagrangean multipliers are chosen to be non-negative ($u \geq 0$), for every optimal solution x of the original optimization problem (P) we have

$$V(\mathrm{LR}_u) \geq V(\mathrm{LR}) \geq V(P)$$

where the operator $V(.)$ denotes the optimal value. Therefore, the resulting objective function from Lagrangean relaxation is an upper bound of the original optimization problem. Let operator Co(.) denotes the convex hull and (P^*) represents the following problem:

$$(P^*) \quad \max \quad z = cx$$
$$\text{s.t. } Ax \leq b$$
$$x \in \mathrm{Co}\{Cx \leq d,\ x \in X\}$$

It can be further shown that (P^*) and (LR) are duals (Geoffrion, 1974). Thus:

$$V(\mathrm{LP}) \geq V(\mathrm{LR}) = V(P^*) \geq V(P)$$

* Corresponding author. Tel.: +1 732 445 2971; fax: +1 732 445 2421.
E-mail address: marianth@sol.rutgers.edu (M. Ierapetritou).

0098-1354/$ – see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2005.12.008

$$(P)\ \max\ z = cx \quad \text{s.t. } Ax \le b,\ Cx \le d,\ x \in X \Longrightarrow (LR_u) = \max\ cx + u(b - Ax) \quad \text{s.t. } Cx \le d,\ x \in X, b \ge 0 \Longrightarrow (LR)\ \min_u (LR_u)$$

Fig. 1. Lagrangean relaxation.

where (LP) is the LP relaxation of problem (P). If the multipliers of the complicating constraints from the optimal LP relaxation solution are used to solve $V(\mathrm{LR}_u)$, the solutions correspond to an upper bound of the original problem at least as tight as the bound from (LP), i.e. $V(\mathrm{LP}) \ge V(\mathrm{LR}_u)$. Geoffrion (1974) presented the following integrality property: the optimal value of (LR_u) is not changed by dropping the integrality condition on the x variables, i.e. $\mathrm{Co}\{Cx \le d,\ x \in X\} = \{Cx \le d,\ x \in X\}$. Only when this property holds, the following equalities hold:

$$V(\mathrm{LP}) = V(\mathrm{LR}) = V(P^*) \ge V(P)$$

However, for practical problems, this integrality property usually does not hold. Thus, Lagrangean relaxation can be applied to provide a tighter bound than LP relaxation.

A drawback of the Lagrangean relaxation however, is that the problem loses its original structure since the complicating constraints are removed from the constraint set and embedded into the objective function. As a method to avoid this, Lagrangean decomposition is presented by Guignard and Kim (1987). It can be regarded as an extension of the Lagrangean relaxation since instead of relaxing constraints it is based on the idea of relaxing a set of variables responsible of connecting important model components. Considering the problem (P) shown in Fig. 2, the set of variables x that connects the set of constrains are duplicated by introducing an identical set of variables y and an equality constraint $x = y$ as shown in Problem (P'). Lagrangean relaxation is then applied to the new set of equality constraints resulting in problem (LD_u), which can be further decomposed into two subproblems (LD_u^1) and (LD_u^2) as shown in Fig. 2.

Let us assume that u° is the optimal Lagrangean multiplier of (LR), $v^\circ = u^\circ A$ and x°, y° are the optimal solution of (LD_u), then:

$$V(\mathrm{LD}_{v^\circ}) = \begin{bmatrix} \max (c - v^\circ)x \\ \text{s.t. } Ax \le b \\ x \in X \end{bmatrix} + \begin{bmatrix} \max v^\circ y \\ \text{s.t. } Cy \le d \\ y \in X \end{bmatrix}$$

$$= \begin{bmatrix} \max (c - u^\circ A)x \\ \text{s.t. } Ax \le b \\ x \in X \end{bmatrix} + \begin{bmatrix} \max u^\circ Ay \\ \text{s.t. } Cy \le d \\ y \in X \end{bmatrix}$$

So the objective function is:

$$= (c - u^\circ A)x^\circ + u^\circ Ay^\circ$$
$$= (c - u^\circ A)x^\circ + u^\circ b + u^\circ(Ay^\circ - b)$$
$$= V(\mathrm{LR}_{u^\circ}) + u^\circ(Ay^\circ - b)$$

Therefore, $V(\mathrm{LD}) \le V(\mathrm{LR})$, which means that the upper bound generated from Lagrangean decomposition is always at least as tight as that from Lagrangean relaxation. Similarly, if the integrality property holds in one of the subproblems, for instance, $\mathrm{Co}\{Cx \le d,\ x \in X\} = \{Cx \le d,\ x \in X\}$, then (LD) is equivalent to (LR) and $V(\mathrm{LD}) = V(\mathrm{LR})$. If integrality property holds in both subproblems, i.e. $\mathrm{Co}\{Ax \le b,\ x \in X\} = \{Ax \le b,\ x \in X\}$ and $\mathrm{Co}\{Cx \le d,\ x \in X\} = \{Cx \le d,\ x \in X\}$, then (LD) is equivalent to (LP) and $V(\mathrm{LD}) = V(\mathrm{LP})$. Otherwise (LD) is strictly tighter than (LR), which is usually the case. The bound of Lagrangean decomposition can be further tightened by adding surrogate constraints, which means both subproblems have overlapping constraints. However, this increases the problem complexity and thus the solution computational time. Fig. 3 provides a geometric interpretation of the Lagrangean decomposition (Guignard & Kim, 1987).

The main difficulty in optimizing Lagrangean decomposition is that it is non-differentiable since the inner maximization prob-

$$P: \begin{bmatrix} \max\ z = cx \\ \text{s.t. } Ax \le b \\ Cx \le d \\ x \in X \end{bmatrix} \Longleftrightarrow P': \begin{bmatrix} \max\ cx \\ \text{s.t. } Ax \le b \\ Cy \le d \\ x = y \\ x \in X,\ y \in X \end{bmatrix} \Longrightarrow LD_u: \begin{bmatrix} \max\ cx + u(y - x) \\ \text{s.t. } Ax \le b \\ Cy \le d \\ x \in X,\ y \in X \end{bmatrix} \Longleftrightarrow$$

$$\left\{ \begin{bmatrix} \max (c - u)x \\ \text{s.t. } Ax \le b \\ x \in X \end{bmatrix}_{LD_u^1} + \begin{bmatrix} \max uy \\ \text{s.t. } Cy \le d \\ y \in X \end{bmatrix}_{LD_u^2} \right\} \Longrightarrow (LD)\ \min_u (LD_u)$$

Fig. 2. Lagrangean decomposition.

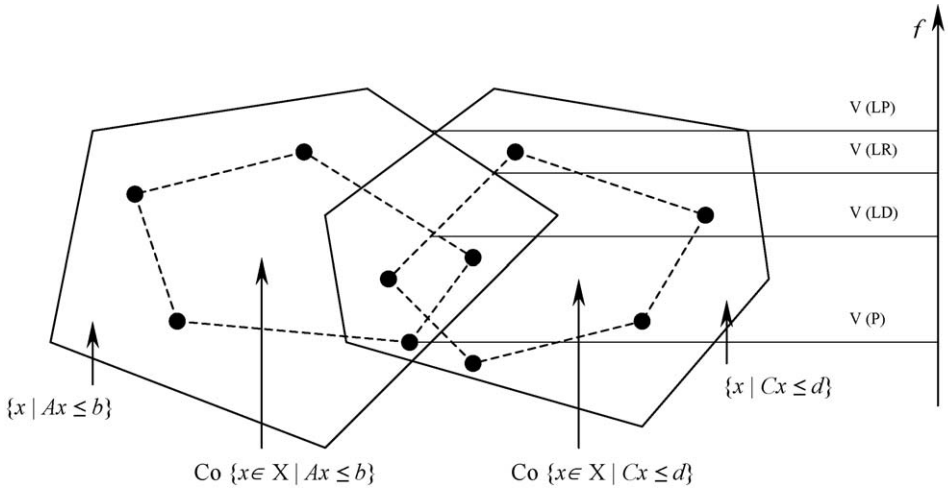

Fig. 3. Geometric interpretation of Lagrangean decomposition.

lem generally has multiple optimal solutions. The most common approach to deal with this lack of differentiability is the subgradient method (Equi, Giorgio, Maziale, & Weintraub, 1997; Marin & Pelegrin, 1998; Polyak, 1967; Rana & Vickson, 1991). The subgradient method utilizes the distance between the objective value at the current iteration Z^k and the estimated optimum Z^* to calculate a step size t_k which is used to update the Lagrangean multipliers as follows:

$$u^{k+1} = u^k + t_k(y^k - x^k) \tag{1}$$

$$t_k = \frac{\lambda(Z^k - Z^*)}{||y^k - x^k||^2}, \quad 0 \le \lambda \le 2 \tag{2}$$

where superscript k corresponds to the iteration number and λ is a scaling factor of the step size to control the convergence, normally considered to be between 0 and 2.

A number of problems however arise with the use of subgradient method. Since Lagrangean decomposition is based on duality theory, theoretically the method converges the dual variables to the same value (more generally, $u(x-y)=0$), which results in the optimal solution of the original optimization problem. However, in practice using subgradient method is reported to result in unpredictable convergence behavior (Guignard, 2003). For some problems, subgradient method generates monotonic improvement in Lagrangean objective function and the convergence is quick and reliable. While other problems result in erratic multiplier sequence and the Lagrangean function value keep deteriorating. According to the complementary slackness condition, subgradient method updates the Lagrangean multiplier u until the convergence of the dual variables is achieved. The gap between the decomposition variables x and y may exist even when the (LD) converges to the optimal objective value of the original optimization problem. We will illustrate this case in the following section through the solution of an example problem. Other drawbacks of subgradient method are (a) the lack of convergence criterion, (b) the lack of estimate of the optimal objective Z^* and (c) dependency to the heuristic choice of the step size sequence (Crowder, 1976). Due to these problems the subgradient method is not stable when applied to large-scale problems. Another issue related to Lagrangean decomposition is that the dualized variable set must be appropriately chosen so that the resulting subproblems are easy to solve and the solution converges fast, which is case-dependent (Gupta & Maranas, 1999; Orero & Irving, 1997).

Bundle method is an extension of subgradient method presented by Lemarechal (1974) and Zowe (1985). This method considers improving the Lagrangean function as well as staying close to the approximated optimal solution. At each iteration, the method either update the Lagrangean multipliers or improves the function approximation. A trade-off exists between the size of the region within which the bundle method allows to move and the bound improvement.

Realizing these deficiencies a number of techniques have been developed to update the Lagrangean multipliers. Constraint generation method (Kelley, 1960) considers a family of k existing solution (x^k, y^k) and generates a new Lagrangean multiplier u by solving a restricted LP master problem (MP^k)

$$\begin{aligned} \min_{u,\eta} \quad & \eta \\ \text{s.t.} \quad & \eta \ge cx^k + u(y^k - x^k), \quad k = 1, \ldots, K \end{aligned}$$

The resulting u is then used in (LD_u) and a new cut of (x^{k+1}, y^{k+1}) is obtained from solving (LD_u), which is added to the master problem (Guignard, 1995). The process terminates when $V(MP^k) = V(LD_u)$. However, there is no guarantee that the new Lagrangean multiplier will generate an improved solution, thus the issue of cycling needs to be resolved. This method also depends on the cuts of (x^k, y^k) in the master problem.

Multiplier adjustment method, also referred as dual ascent/descent algorithm was presented by Bilde and Krarup (1977) and reported successful application by Erlenkotter (1978), Fisher and Hochbaum (1980), Fisher and Kedia (1990) and Guignard and Rosenwein (1990). This method generates a

sequence of u^k by using the following relationship:

$$u^{k+1} = u^k + t^k d^k \quad (3)$$

where t^k is a positive scalar and d^k is a descent direction. d^k is usually determined from a finite set of directions by evaluating the directional derivative of (LD_u). Typically the direction of the steepest descent is chosen and the step size t^k minimizes $V(LD_u k_{+td} k)$. Although this method is reported to work better than the subgradient method for some problems (Erlenkotter, 1978), the set of directions to choose from may involve specific knowledge of the problem in order to reduce the set of directions examined. A good review of the methods solving for Lagrangean multipliers is given by Guignard (2003).

These issues initiate our efforts towards an improved method for updating the Lagrangean multipliers based on direct search in Lagrangean multiplier space. In the next section a motivating example illustrates the need for an improved updating method for the Lagrangean multipliers. In Section 3, the proposed modified Lagrangean decomposition method is presented whereas examples are given in Section 4.

2. Motivating example

The following mixed integer linear programming problem is considered in Wu and Ierapetritou (2003).

$$\max 2x_1 + 3x_2 + 4x_3 + 4x_4$$
$$s.t.$$
$$x_1 + 2x_2 \le 8 \quad (1)$$
$$4x_2 + 3x_3 \le 13 \quad (2)$$
$$2x_1 + 5x_4 \le 11 \quad (3)$$
$$x_3 + x_4 \le 6 \quad (4)$$
$$x_1, x_2, x_3, x_4 \in Z^+ \cup \{0\}$$

The optimal objective function value is 26 that corresponds to two equivalent solutions $(x_1, x_2, x_3, x_4) = (5, 0, 4, 0)$ and $(x_1, x_2, x_3, x_4) = (3, 0, 4, 1)$. Subgradient method is used for the solution of this problem with the initial value of λ equal to 2. The updating strategy halves λ when the objective function is not improving for five iterations. As described in previous section, the choice of the decomposing variables is case dependent and affects the convergence of the Lagrangean decomposition. Table 1 shows the results for this example when different variables are used for the decomposition.

Although the Lagrangean decomposition converges to the optimal solution, the primary variables are not guaranteed to be optimal. For example, the Lagrangean decomposition returns the optimal objective function value of 26 when x_2 and x_3 are decomposed, the solution of Lagrangean decompositon however, corresponds to $(x_2, x_3) = (0, 6)$ whereas $(y_2, y_3) = (0, 4)$. The subgradient method terminates since $Z^k = Z^*$. Therefore, we obtain an upper bound, which converges to the optimal objective function of the original problem, but the variables do not converge to the optimal values and a gap between the dualized variables exists. When x_1 and x_2 are used for decomposition as follows:

$$\max 4x_3 + 4x_4 + (2 - u_1)x_1 + (3 - u_2)x_2$$
$$\text{s.t.} \qquad + \qquad \max u_1 y_1 + u_2 y_2$$
$$4x_2 + 3x_3 \le 13 \qquad \text{s.t.}$$
$$2x_1 + 5x_4 \le 11 \qquad y_1 + 2y_2 \le 8$$
$$x_3 + x_4 \le 6 \qquad y_1, y_2 \in Z^+ \cup \{0\}$$
$$x_1, x_2, x_3, x_4 \in Z^+ \cup \{0\}$$

Lagrangean decomposition converges to a suboptimal solution of 26.012. This is due to the fact that when the solution of $(x_1, x_2, x_3, x_4, y_1, y_2) = (5, 0, 4, 0, 0, 4)$ is generated, the duality gaps of x_1,y_1 and x_2,y_2 forces u_1 and u_2 in the area where the same solutions are generated until the step size becomes too small and the algorithm terminates. Table 2 lists the details for each iteration.

Moreover the performance of Lagrangean decomposition depends on the value of Z^* used as shown in Table 3 for the case where $Z^* = 27$. If x_3 and x_4 are dualized, the Lagrangean decomposition results in an objective function value of 26.462 compared to the optimal value of 26 when $Z^* = 26$ is used (Table 1). In the same example when $Z^* = 25$ is used, the Lagrangean decomposition converges to the optimal value of 26 as the lowest upper bound but requires 85 iterations compared to four iterations

Table 2
Intermediate results from iterations

Iteration	$(x_1, x_2, x_3, x_4, y_1, y_2)$	(u_1, u_2)	t^k	(LD_u)
1	(0, 0, 4, 2, 8, 0)	(1.000, 1.000)	1.0000	32.000
2	(5, 0, 4, 0, 0, 4)	(0.000, 1.000)	0.1875	30.000
3	(0, 0, 4, 2, 8, 0)	(0.976, 0.220)	0.1951	31.805
4	(5, 0, 4, 0, 0, 4)	(0.000, 0.220)	0.1814	26.878
5	(5, 0, 4, 0, 8, 0)	(0.214, 0.048)	0.0428	26.642
6	(5, 0, 4, 0, 0, 4)	(0.000, 0.048)	0.1428	26.193
7	(5, 0, 4, 0, 8, 0)	(0.024, 0.029)	0.0047	26.071
8	(5, 0, 4, 0, 0, 4)	(0.000, 0.029)	0.0078	26.118
...	...	...	...	...
32	(5, 0, 4, 0, 0, 4)	(0.004, 0.008)	0.0000	26.012

Table 1
Subgradient method results of motivating example

Decomposition variables	Optimal solution (x_1, x_2, x_3, x_4)	Dual variables	Dual variable values	u_1, u_2	Optimal objective function value using subgradient method	CPU (s)
x_1, x_2	(5,0,4,0)	y_1, y_2	(0,4)	(0.00,0.00)	26.012	1.19
x_2, x_3	(5,0,6,0)	y_2, y_3	(0,4)	(3.37, 4.00)	26.000	0.27
x_1, x_4	(0,0,4,0)	y_1, y_4	(3,1)	(2.00, 4.37)	26.279	2.40
x_3, x_4	(5,0,4,0)	y_3, y_4	(6,0)	(0.00, 0.00)	26.000	0.12

Table 3
Lagrangean decomposition results based on estimation of $Z^* = 27$

Decomposing variables	Optimal objective function value using sub-gradient method
x_1, x_2	26.816
x_2, x_3	26.500
x_1, x_4	27.000
x_3, x_4	26.462

Table 4
Lagrangean decomposition results based on estimation of $Z^* = 25$

Decomposing variables	Optimal objective function value using sub-gradient method
x_1, x_2	26.000
x_2, x_3	26.000
x_1, x_4	26.000
x_3, x_4	26.000

required when $Z^* = 26$ is used. Table 4 illustrates the results for $Z^* = 25$.

The dependency on the dual variables and Z^* is also observed with initial λ equal to 1 when subgradient method is used to update the Lagrangean multipliers, therefore the need of alternative methodology becomes imperative for the efficient utilization of Lagrangean decomposition in large-scale problem describing realistic case studies.

3. Proposed approach

The proposed approach uses a direct search method to update the Lagrangean multipliers in order to improve the performance of the Lagrangean decomposition. The main idea is that given a fairly good estimation of Lagrangean multipliers, only the promising directions need to be searched. Thus, the computational complexity decreases and the objective of the Lagrangean decomposition is improved at each iteration.

Nelder–Mead method (Nelder & Mead, 1965) is used to determine the promising search directions since it is proven to be a very efficient direct search algorithm. For a function of n variables, the algorithm maintains a set of $n+1$ points forming the vertices of a simplex or polytope in n-dimensional space. The result of each iteration is either a new vertex for the next iteration which replaces the one having the worst objective function value, or if a shrink is performed, a set of $n+1$ new points forming the simplex for the next iteration. Four scalar parameters must be specified to define a complete Nelder–Mead method, which are the coefficients of reflection, expansion, contraction, and shrinking. As for every direct search method Nelder–Mead method has the advantage of not requiring derivative computations. However, due to computational complexity of the algorithm it tends to be efficient in relatively low dimensions. The details of the Nelder–Mead algorithm are given in the appendix. As shown in Fig. 4, the approach starts with a set of $n+1$ points for an n dimensional problem. The $n+1$ points form the vertices of a simplex or polytope in the n-dimensional space. This simplex is successively updated by expansion, contraction or multiple contraction, at each iteration by discarding the vertex having the worst function value and replacing it with a new vertex having a best function value.

In order to be able to efficiently use the Nelder–Mead method we need to determine a good initial set of Lagrangean multipliers to reduced the search space. Moreover, the promising search

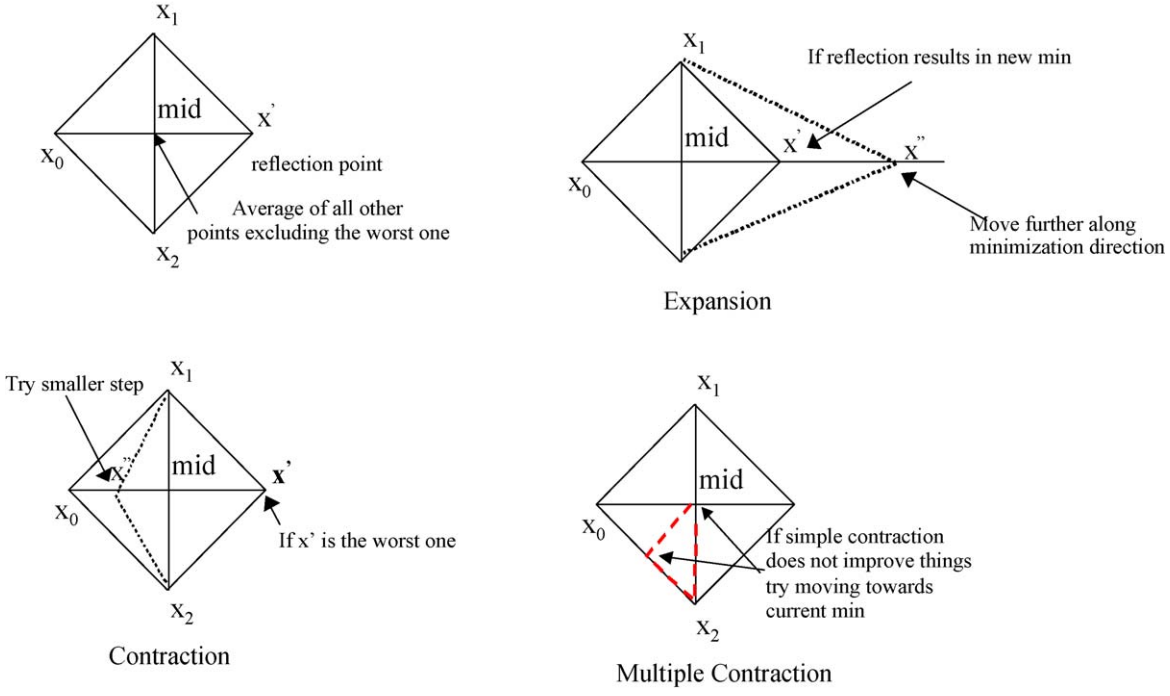

Fig. 4. Illustration of the basic functions of Nelder–Mead Method.

directions should be easily determined. These two questions are addressed as follows. As a common practice, an effective way to generate an initial set of Lagrangean multipliers for integer linear problems is to use the dual values of LP relaxation. This is because the Lagrangean decomposition problem is equivalent to LP relaxation if we drop the integrality constraints, thus most of these values are already near the optimum and we only need to examine the reduced Lagrangean multiplier space for the next update. The second question is most critical to the proposed algorithm. Promising search direction is defined as the direction resulting in improvement of the objective function. In order to adjust the Lagrangean multipliers independently, the axes in the Lagrangean multiplier space are used as the possible directions. Therefore, we have n orthogonal directions to search where n is the number of Lagrangean multipliers and at each iteration we should choose among the directions that lead to the objective function improvement. In practice the directions leading to objective function improvement are much less than the actual number of Lagrangean multipliers because of their good initial values.

The steps of the proposed approach are shown in Fig. 5. In particular first the LP relaxation of the original optimization problem is solved and the Lagrangean multipliers are initialized as the dual values of the corresponding equality constraints. Assuming that there are n pairs of dual variables, the Lagrangean multiplier space thus has n dimensions. To generate the initial points for Nelder–Mead algorithm, we fix the $n-1$ variables at their original values and change only one dimension by a value of $\pm\Delta u$, which is α times the original value $\Delta u = \alpha^* u_0$. In this way two new points are generated. We apply the same procedure for all dimensions and generate $2n$ points. Each point corresponds to a set of Lagrangean multipliers. The next step is to solve $2n+1$ Lagrangean decomposition problems associated with these points and sort them based on their objective function values. Although this might be a time-consuming step, a potential advantage is that such multi-direction search can be computed in parallel since these problems are completely independent of each other. Those with improved objective function values are selected to form the reduced space where Nelder–Mead algorithm is applied. The resulted best point is guaranteed to be at least as good as the previous best point, and thus it is used as the new starting point to generate the next $2n$ neighboring points. The iterations continue until the difference in the objective function values is within tolerance or the number of iterations reaches a limit.

Since each Nelder–Mead iteration returns a new point with equal or better objective function than the previous point, convergence is guaranteed. However, the algorithm's efficiency depends on the value α. A large value of α gives more emphasis in the most promising directions in Lagrangean multiplier space and results in larger changes; while small value of α concentrates on limited areas and attempts to find all promising directions although it may result in slower convergence. An adaptive strategy is thus proposed starting with a value of α at the initial iterations in order to improve the values of the Lagrangean multiplier faster, and reducing α when no new improving directions can be found. This strategy will be illustrated in the following section where first the moti-

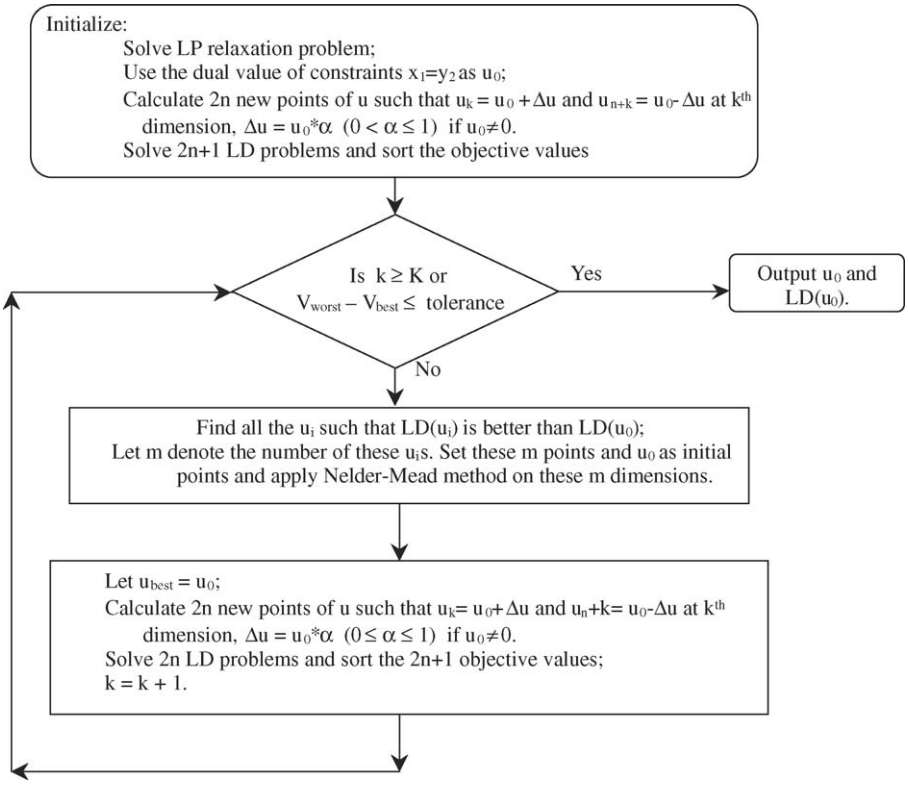

Fig. 5. Proposed approach for Lagrangean decomposition using Nelder–Mead algorithm.

Table 5
Comparison of results and CPU times for motivating example

Dual variables	Subgradient method		Nelder–Mead method		Proposed algorithm	
	Objective	CPU (s)	Objective	CPU (s)	Objective	CPU (s)
x_1, x_2	26.012	1.19	26.000	0.10	26.000	0.10
x_2, x_3	26.000	0.27	26.000	11.07	26.000	0.33
x_1, x_4	26.279	2.40	26.000	6.30	26.000	0.73
x_3, x_4	26.000	0.12	26.000	0.10	26.000	0.10

Table 6
Solution of motivating example

Decomposing variables	(x_1, x_2, x_3, x_4)	u_1, u_2	Dual variables	Dual variable values	Iterations
x_1, x_2	(5, 0, 4, 0)	(0.00, 0.00)	y_1, y_2	(0,0)	5
x_2, x_3	(5, 0, 0, 0)	(3.00, 4.00)	y_2, y_3	(0,4)	21
x_1, x_4	(0, 0, 4, 0)	(2.00, 4.00)	y_1, y_4	(5,0)	34
x_3, x_4	(5, 0, 4, 0)	(0.00, 0.00)	y_3, y_4	(0,0)	5

vating example of Section 2 is revisited using the proposed approach and the performance is compared with the subgradient method. An additional example and case studies in the area of process operation are considered to illustrate the importance of the proposed approach in the solution of large-scale problems.

4. Case studies

4.1. Motivating example

Unlike the subgradient method, the proposed approach based on Nelder–Mead algorithm is not sensitive to the selection of the decomposition variables since it has the advantage of searching for the optimal direction in the Lagrangean multiplier space. As shown in Table 5 the proposed algorithm converges to the optimal solution despite of the choice of the decomposing variables. The CPU time in most of the cases is much better than that of the subgradient method. The results are obtained on Pentium 1200 PC using GAMS/CPLEX 7.5.

To further investigate the performance of the proposed approach the same strategy is implemented but using the standard Nelder–Mead algorithm. The results are shown in Table 5. As expected the proposed modified algorithm performs better since it investigates only the promising search directions.

The optimal solutions in terms of primal and dual variables, Lagrangean multipliers and number of iterations of the proposed algorithm are listed in Table 6. The α parameter is fixed to 0.1% of the original values. When α is equal to 1 and 0.05%, the algorithm converges to the same solution, however, the corresponding numbers of iterations (Table 7) are different as discussed in the previous section.

4.2. Example 1

Guignard and Kim (1987) used the following example to illustrate that the Lagrangean decomposition generates a bound at least as good as Lagrangean relaxation:

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 + 4x_3 \\ \text{s.t.} \quad & 12x_1 + 19x_2 + 30x_3 \leq 46 \\ & 49x_1 + 40x_2 + 31x_3 \leq 76 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{aligned}$$

In this example, the optimal objective value is 4 with the solution of $\{x_1, x_2, x_3\} = \{0, 0, 1\}$. The authors stated that with Lagrangean multipliers of $u_1 = 2$, $u_2 = 0.5$ and $u_3 = 1.5$, Lagrangean decomposition can achieve an upper bound of 4.5, tighter than the upper bound of 5.84 obtained by Lagrangean relaxation. Using the LP relaxation to initialize Lagrangean multipliers of u_1, u_2, u_3, the subgradient method converges to the optimal value of 4.5 in 39 iterations; while the proposed approach reduces the number of iterations by 25% (29 iterations versus 39 iterations). The α parameter is set to an initial value of 1 and reduced to 60% of its value when promising new directions cannot be obtained.

4.3. Scheduling problem

A problem of short-term scheduling of multi-product, multipurpose batch plants is considered in this case study. Although continuous-time representation reduces the size of the model as emphasized by Ierapetritou and Floudas (1998), it is still difficult to solve the realistic problems to optimality. Therefore, decomposition-based techniques have been studied to reduce the computational complexity of the solution approach and enable the utilization of optimization techniques to industrial size prob-

Table 7
Number of iterations using different α

Decomposing variables	$\alpha = 1\%$	$\alpha = 0.1\%$	$\alpha = 0.05\%$
x_1, x_2	5	5	5
x_2, x_3	13	21	21
x_1, x_4	20	34	39
x_3, x_4	5	5	5

lems. The proposed Lagrangean decomposition approach is used here for an example problem involving five processes, nine raw materials and two products. The data for the problem are provided in Table 8. The schedule considered in this section spans over three time periods of 8, 8 and 12 h, respectively. The objective is to maximize the total profit in these periods. For each of the three subproblems corresponding to different period, continuous time formulation is used to determine the optimal sequence of tasks and details of efficient scheduling. For completeness in the presentation the scheduling model involves the following constraints:

$$\sum_{i \in I_j} wv(i, n_k) = yv(j, n_k), \quad \forall j \in J,\ n_k \in N_k$$

$$V_{ij}^{\min} wv(i, n_k) \le B(i, j, n_k) \le V_{ij}^{\max} wv(i, n_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k$$

$$\mathrm{ST}(s, n_k) \le \mathrm{ST}_s^{\max}, \quad \forall s \in S,\ n_k \in N_k$$

$$\mathrm{ST}(s, n_k) = \mathrm{ST}(s, n_k - 1) - d(s, n_k) + \sum_{i \in I_s} \rho_{si}^{p} \sum_{j \in J_i} B(i, j, n_k - 1) - \sum_{i \in I_s} \rho_{si}^{c} \sum_{j \in J_i} B(i, j, n_k), \quad \forall s \in S,\ n_k \in N_k$$

$$\mathrm{ST}(s, n_k) = \mathrm{STIN} - d(s, n_k) - \sum_{i \in I_s} \rho_{si}^{c} \sum_{j \in J_i} B(i, j, n_k), \quad \forall s \in S,\ n_k \in N_k,\ n_k = 1$$

$$\sum_{n_k \in N_k} d(s, n) \ge r_s, \quad \forall s \in S$$

$$T^f(i, j, n_k) = T^s(i, j, n_k) + \alpha_{ij} wv(i, n_k) + \beta_{ij} B(i, j, n_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k$$

$$T^s(i, j, n_k + 1) \ge T^f(i, j, n_k) - H(2 - wv(i, n_k) - yv(j, n_k)), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ n_k \ne N_k$$

$$T^s(i, j, n_k + 1) \ge T^s(i, j, n_k), \quad \forall i \in I,\ j \in J_i,\ n \in N,\ n_k \ne N_k$$

$$T^f(i, j, n_k + 1) \ge T^f(i, j, n_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ n_k \ne N_k$$

$$T^s(i', j, n_k + 1) \ge T^f(i, j, n_k) - H(2 - wv(i', n_k) - yv(j, n_k)), \quad \forall j \in J,\ i \in I_j,\ i' \in I_j,\ i \ne i',\ n_k \in N_k,\ n_k \ne N_k$$

$$T^s(i', j', n_k + 1) \ge T^f(i, j, n_k) - H(2 - wv(i', n_k) - yv(j', n_k)), \quad \forall j,\ j' \in J,\ i_j,\ i' \in I_j,\ i \ne i',\ n_k \in N,\ n_k \ne N_k$$

$$T^s(i, j, n_k + 1) \ge \sum_{n_k' \in N_k, n_k' \le n_k} \sum_{i' \in I_j} (T^f(i', j, n_k') - T^s(i', j, n_k')), \quad \forall i \in I,\ j \in J_i,\ i \ne i',\ n_k \in N_k,\ n \ne N_k$$

$$T^f(i, j, n_k) \le H_k, \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k$$

$$T^s(i, j, n_k) \le H_k, \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k$$

$$\max \sum_k \sum_s \sum_{n_k} \mathrm{price}(s) \times d(s, n_k)$$

The first two constraints in the above formulation express the allocation of tasks to units and the requirement of the batch sizes to be within the capacity limits of the units. The next three constraints maintain the material balance for all materials involved and that the demand requirements are satisfied, whereas the timing constraints enforce the correct duration and sequence of tasks to align with the recipe of different products. Although the above formulation has been proved to perform very well (Ierapetritou & Floudas 1998), the simultaneous solution of this problem for all three periods is computationally expensive due to the large number of binary variables involved. With Lagrangean decomposition, the problem is decomposed into three subproblems by dualizing the storage variables at the end of each time period. Note that the selection of this decomposition strategy is imposed by the nature of the problem since it provides three separable subproblems.

Both subgradient method and the proposed algorithm are utilized to obtain an upper bound whereas a lower bound is generated by fixing the binary variables of the first period from the Lagrangean decomposition solution. The results are obtained on Sun Ultra 60 workstation (360 Mhz Ultra-SparcII Prosessors with 1 Gb of Memory) using GAMS/CPLEX 7.5. Table 9 compares the results of simultaneous solution, subgradient method

Table 8
Data for scheduling problem

Unit	Capacity	Suitability	Mean processing time ($\tau_{ij}^{\mathrm{mean}}$)
Heater	100	Heating	1.0
Reactor 1	50	Reaction 1–3	2.0, 2.0, 1.0
Reactor 2	80	Reaction 1–3	2.0, 2.0, 1.0
Still	200	Separation	2.0

State	Storage capacity	Initial amount	Price
Feed A	Unlimited	Unlimited	0.0
Feed B	Unlimited	Unlimited	0.0
Feed C	Unlimited	Unlimited	0.0
Hot A	100	0.0	0.0
Int AB	200	0.0	0.0
Int BC	150	0.0	0.0
Impure E	200	0.0	0.0
Product 1	Unlimited	0.0	10.0
Product 2	Unlimited	0.0	10.0

Table 9
Comparisons for the scheduling example

	Simultaneous solution	Lagrangean decomposition with subgradient method	Lagrangean decomposition with the proposed algorithm
Upper bound (LD Obj.)	7220.79[a]	7185.78	7176.11
Lower bound (feasible schedule)	7067.90	7115.31	7115.31
Relative gap (%)	2.16	0.99	0.85

[a]The upper bound provided by the solver at the time of termination.

and the proposed algorithm for a maximally allowable computation time of 10,000 s.

For maximum computation time of 20,000 s, the proposed approach results in an improved upper bound of 7174.93 and lower bound of 7165.74 with 0.13% gap, while the simultaneous solution and subgradient method result in no improvement in the solution.

4.4. Scheduling under uncertainty

Uncertainty receives lots of attention in the research of scheduling and planning problems. A commonly used approach is a multi-stage optimization that utilizes a set of scenarios to represent the possible parameter values (Dantzig, 1955). This however, increases the problem size due to the scenarios introduced. Lagrangean decomposition is thus an ideal technique to use in order to efficiently address the problem of scheduling under uncertainty.

The problem considered here involves four different products produced through eight processing tasks. The State Task Network (STN) representation for this example can be found in Wu and Ierapetritou (2003) as Example 2. The data are provided in Table 10. Two periods are considered, the first period of 6 h where the parameters are assumed deterministic and the second period of 6 h where uncertainty is considered. In this example, three scenarios are used in the second period to describe the uncertain demands and prices corresponding to high, medium and low as shown in Table 11. The objective function combines the maximization of profit for both periods while satisfying the market demands, and thus determines the optimal production schedule for the first period that will benefit the entire time horizon under consideration.

The mathematical formulation for the second period is similar to that in the previous example, however it considers multiple scenarios as shown below:

$$\sum_{i \in I_j} wv(i, n_k, q_k) = yv(j, n_k, q_k), \quad \forall j \in J,\ n_k \in N_k,\ q_k \in Q$$

$$V_{ij}^{\min} wv(i, n_k, q_k) \le B(i, j, n_k, q_k) \le V_{ij}^{\max} wv(i, n_k, q_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ q_k \in Q$$

$$\mathrm{ST}(s, n_k, q_k) \le \mathrm{ST}_s^{\max}, \quad \forall s \in S,\ n_k \in N_k,\ q_k \in Q$$

$$\sum_{n_k \in N_k} d(s, n, q_k) \ge r_{s,q_k}, \quad \forall s \in S,\ q_k \in Q$$

$$\mathrm{ST}(s, n_k, q_k) = \mathrm{ST}(s, n_k - 1, q_k) - d(s, n_k, q_k) + \sum_{i \in I_s} \rho_{si}^{p} \sum_{j \in J_i} B(i, j, n_k - 1, q_k) - \sum_{i \in I_s} \rho_{si}^{c} \sum_{j \in J_i} B(i, j, n_k),$$

$$\forall s \in S,\ n_k \in N_k,\ q_k \in Q$$

$$\mathrm{ST}(s, n_k, q_k) = \mathrm{STIN} - d(s, n_k, q_k) - \sum_{i \in I_s} \rho_{si}^{c} \sum_{j \in J_i} B(i, j, n_k, q_k), \quad \forall s \in S,\ n_k \in N_k,\ n_k = 1,\ q_k \in Q$$

$$T^f(i, j, n_k, q_k) = T^s(i, j, n_k, q_k) + \alpha_{ij} wv(i, n_k, q_k) + \beta_{ij} B(i, j, n_k, q_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ q_k \in Q$$

$$T^s(i, j, n_k + 1, q_k) \ge T^f(i, j, n_k, q_k) - H(2 - wv(i, n_k, q_k) - yv(j, n_k, q_k)), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ n_k \ne N_k,\ q_k \in Q$$

$$T^s(i, j, n_k + 1, q_k) \ge T^s(i, j, n_k, q_k), \quad \forall i \in I,\ j \in J_i,\ n \in N,\ n_k \ne N_k,\ q_k \in Q$$

$$T^f(i, j, n_k + 1, q_k) \ge T^f(i, j, n_k, q_k), \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ n_k \ne N_k,\ q_k \in Q$$

$$T^s(i', j, n_k + 1, q_k) \ge T^f(i, j, n_k, q_k) - H(2 - wv(i', n_k, q_k) - yv(j, n_k, q_k)), \quad \forall j \in J,\ i \in I_j,\ i' \in I_j,\ i \ne i',\ n_k \in N_k,\ n_k \ne N_k,\ q_k \in Q$$

$$T^s(i', j', n_k + 1, q_k) \ge T^f(i, j, n_k, q_k) - H(2 - wv(i', n_k, q_k) - yv(j', n_k, q_k)), \quad \forall j,\ j' \in J,\ i_j,\ i' \in I_j,\ i \ne i',\ n_k \in N_k,\ n_k \ne N_k,\ q_k \in Q$$

$$T^s(i, j, n_k + 1, q_k) \ge \sum_{n_k' \in N_k, n_k' \le n_k i' \in I_j} \sum (T^f(i', j, n_k', q_k) - T^s(i', j, n_k', q_k)), \quad \forall i \in I,\ j \in J_i,\ i \ne i',\ n_k \in N_k,\ n \ne N_k,\ q_k \in Q$$

$$T^f(i, j, n_k, q_k) \le H_k, \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ q_k \in Q$$

$$T^s(i, j, n_k, q_k) \le H_k, \quad \forall i \in I,\ j \in J_i,\ n_k \in N_k,\ q_k \in Q$$

$$\max \sum_{k} \sum_{s} \sum_{n_k} \sum_{q_k} \mathrm{weight}\,(q_k) \times \mathrm{price}(s, q_k) \times d(s, n_k, q_k)$$

Table 10
Data for scheduling problem under uncertainty

Unit	Capacity	Suitability	Mean processing time (τ_{ij}^{mean})
Unit 1	1000	Task 1	1.0
Unit 2	2500	Tasks 3 and 7	1.0
Unit 3	3500	Task 4	1.0
Unit 4	1500	Task 2	1.0
Unit 5	1000	Task 6	1.0
Unit 6	4000	Tasks 5 and 8	1.0

State	Storage capacity	Initial amount	Price
Feeds 1–3	Unlimited	0.0	0.0
Intermediate 4	1000	0.0	0.0
Intermediate 5	1000	0.0	0.0
Intermediate 6	1500	0.0	0.0
Intermediate 7	2000	0.0	0.0
Intermediate 8	0	0.0	0.0
Intermediate 9	3000	0.0	0.0
Products 1–4	Unlimited	0.0	18–21
Feeds 1–3	Unlimited	Unlimited	0.0

Table 11
Demand and price in the second period

Product	Scenario 1		Scenario 2		Scenario 3	
	Demand	Price	Demand	Price	Demand	Price
P1	800	15	1800	18	2400	21
P2	3000	16	4500	19	3800	20
P3	500	24	700	20	600	15
P4	3000	25	2500	21	2000	18

The mathematical formulation corresponds to a MILP, which was solved on Sun Ultra 60 workstation (360 Mhz Ultra-SparcII Prosessor with 1 Gb of Memory) using GAMS/CPLEX 7.5. The branch-and-bound algorithm is not efficient for this problem as shown by the results presented in Table 12. We thus applied the proposed Lagrangean decomposition approach for solving this problem. First the original planning problem is decomposed into four subproblems by dualizing the storage variables between the first and second period. The objective function of Lagrangean decomposition corresponds to an upper bound of the original optimization problem while a lower bound is generated by fixing the binary variables of the first period at the values of the Lagrangean decomposition solution and solving the original problem. Table 12 compares the objective value, upper bound and relative gap obtained using the proposed approach and solving the original problem using branch-and-bound.

Table 12
Results of the scheduling problem with uncertainty

	Directly solving the problem	Proposed Lagrangean decomposition approach
Iteration	7	2
CPU (s)	30,000[a]	17,995
Objective function value	4.131E5	4.148E5
Upper bound	4.740E5[b]	4.329E5
Relative gap	14.7%	4.4%

[a] Computational time limit.
[b] The upper bound provided by the solver at the time of termination.

It should be pointed out that the upper bound is improving with the number of iterations. After nine iterations the upper bound is further reduced to 4.3E5 with a relative gap of 3.7%. However, we need to balance the tradeoff between the objective function improvement and the computational efficiency. Note that for this problem the subgradient method could not get an improved solution after the first iteration resulting in a Lagrangean decomposition value of 4.35E5.

5. Summary

This paper presents a Lagrangean decomposition scheme using a Nelder–Mead algorithm to search for promising directions in Lagrangean multiplier space. At each iteration the proposed method provides a solution at least as good as that of the previous iteration, thus enabling better convergence. Focusing only on promising directions prevents the algorithm from evaluating large number of points, which is the main disadvantage of direct-search algorithms. The case studies illustrate the algorithm can be utilized as an alternative for the problems where subgradient method performs poorly. It can also be employed in combination with subgradient method in a scheme where this method was only utilized when subgradient stops improving the objective function. Special emphasis has been placed in the performance of the proposed approach to scheduling problems for which promising results are obtained.

Acknowledgement

The authors would like to acknowledge the financial support from NSF (CTS-0224745).

Appendix A. Nelder–Mead method (Kelley, 1999)

Nelder-Mead Method (Kelley, 1999)

Initialize:

m+1 points for m dimensions sorted according to $LD(u_i)$, i.e. $LD(u_0) \le LD(u_1) \le \ldots\ldots \le LD(u_m)$.

Main:

While $kc \le L$ and $LD(u_m) - LD(u_0) \ge$ tolerance, do

Compute $u_{bar}=\sum_{i=1}^{m} u_i/m$;

Reflect: $u_r = u_{bar} + \lambda_r(u_{bar}-u_m)$;
Compute $LD(u_r)$;

If ($LD(u_0) \le LD(u_r) < LD(u_{m-1})$), do
Replace u_m with u_r ;
Else if ($LD(u_r) < LD(u_0)$), do
Expand: $u_e = u_r + \lambda_e(u_r-u_{bar})$;
Compute $LD(u_e)$;

If $LD(u_e) < LD(u_r)$, do
Replace u_m with u_e ;
Else do
Replace u_m with u_r ;
Else if ($LD(u_{m-1}) \le LD(u_r) < LD(u_m)$), do
Outside Contraction: $u_b = u_{bar} + \lambda_b(u_{bar}-u_m)$;
Compute $LD(u_b)$;
If $LD(u_b) < LD(u_r)$, do
Replace u_m with u_b ;
Else do
Shrink: $u_i = u_0 - \lambda_s(u_i-u_0)\ \forall i \in \{1,\ldots,m\}$;
Compute $LD(u_i)\ \forall i \in \{1,\ldots,m\}$;
End do
Else if ($LD(u_m) \le LD(u_r)$), do
Inside Contraction: $u_c = u_{bar} - \lambda_c(u_{bar}-u_m)$;
Compute $LD(u_c)$;
If $LD(u_c) < LD(u_m)$, do
Replace u_m with u_c ;
Else do
Shrink: $u_i = u_0 - \lambda_s(u_i-u_0)\ \forall i \in \{1,\ldots,m\}$;
Compute $LD(u_i)\ \forall i \in \{1,\ldots,m\}$;
End do
End do
Sort: Sort u_i such that $LD(u_0) \le LD(u_1) \le \ldots\ldots \le LD(u_m)$.
$kc = kc + 1$.
End do

Output: u_0 and $LD(u_0)$

$\{\lambda_r, \lambda_e, \lambda_b, \lambda_c, \lambda_s\} = \{1, 1, 1/2, 1/2, 1/2\}$ (recommended)

References

Bilde, & Krarup. (1977). Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals Discrete Mathematics*, *1*, 79–97.

Benders, J. F. (1962). Partitioning procedures for solving mixed variables programming problems. *Numberische Mathematik*, *4*, 238.

Crowder, H. (1976). *Computational improvements for subgradient optimization*. *Symposia mathematica: vol. XIX*. London: Academic Press.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton, NJ: Princeton University Press.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, *1*, 197–206.

Diaby, M., Bahl, H. C., Karwan, M. H., & Zionts, S. (1992). A Lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Science*, *9*, 1329.

Equi, L., Giorgio, G., Maziale, S., & Weintraub, A. (1997). A combined transportation and scheduling problem. *European Journal of Operational Research*, *97*, 94.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operational Research*, *26*(1), 992–1009.

Fisher, M. L. (1973). Optimal solution of scheduling problems using lagrange multipliers: Part I. *Operational Research*, *21*, 1114–1127.

Fisher, M. L., & Hochbaum, D. S. (1980). Database location in a computer network. *Journal of ACM*, *27*, 718–735.

Fisher, M. L. (1981). The Lagrangean relaxation method for solving integer programming problems. *Management Science*, *27*, 1.

Fisher, M. L., & Kedia, P. (1990). Optimal solution of set sconvering/partitioning problems using dual heuristics. *Management Science*, *39*, 67–88.

Geoffrion, A. M. (1974). Lagrangean relaxation and its uses in integer programming. *Mathematics Programming Studies*, *2*, 82.

Graves, S. C. (1982). Using Lagrangean techniques to solve hierarchical production planning problems. *Management Science*, *28*, 260.

Guignard, M., & Kim, S. (1987). Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Program*, *39*, 215–228.

Guignard, M., & Rosenwein, M. B. (1990). An application of Lagrangean decomposition to the resource-constrained minimum weighted arborescence problem. *Networks*, *20*, 345–359.

Guignard, M. (1995). Lagrangean relaxation: A short course. *Belgian Journal of Operational Research*, *35*(3–4), 95.

Guignard, M. (2003). Lagrangean relaxation. *Top*, *11*(2), 151–228.

Gupta, A., & Maranas, C. (1999). A hierarchical Lagrangean relaxation procedure for solving midterm planning problems. *Indian Engineering of Chemical Research*, *38*, 1937–1947.

Held, M., & Karp, R. M. (1970). The traveling salesman problem and minimum spanning trees. *Operational Research*, *18*, 1138–1162.

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Indian Engineering of Chemical Research*, *37*, 4341.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of SIAM*, *8*, 703–712.

Kelley, C. T. (1999). *Iterative methods for optimization*. Philadelhia: Society for Industrial and Applied Mathematics.

Lemarechal, C. (1974). An Algorithm for Minimizing Convex Functions. In *Proceedings IFIP'74 Congress* (pp. 552–556). North Holland.

Marin, A., & Pelegrin, B. (1998). The return plant location problem: Modeling and resolution. *European Journal of Operational Research*, *104*, 375.

Nelder, J. A., & Mead, R. (1965). A Simplex method for function minimization. *Computer Journal*, 308–313.

Orero, S. O., & Irving, M. R. (1997). A combination of the genetic algorithm and Lagrangian relaxation decomposition techniques for the generation unit commitment problem. *Electric Power System Research*, *43*, 149–156.

Polyak, B. T. (1967). A general method of solving extremum problems. *Soviet Mathematics Doklady*, *8*, 593–597.

Rana, K., & Vickson, R. G. (1991). Routing container ships using Lagrangean relaxation and decomposition. *Transaction Science*, *25*(3).

Thizy, J. M., & Wassenhove, L. N. V. (1985). Lagrangean relaxation for the multi-iterm capacitated lot-sizing problem: A heuristic implementation. *IIE Transaction*, *17*, 308.

Wu, D., & Ierapetritou, M. G. (2003). Decomposition approaches for the efficient solution of the short-term scheduling problems. *Computational Chemical Engineering*, *27*, 1261–1276.

Zowe, J. (1985). Nondifferentiable optimization. In K. Schittkowski, *Computational Mathematical Program, NATO ASI Series F: Computer and Systems Science* (vol. 15, pp. 323–356). Springer-Verlag.