PERGAMON

Computers & Operations Research 28 (2001) 33–52

computers & operations research

www.elsevier.com/locate/dsw

A modified subgradient algorithm for Lagrangean relaxation

Francesca Fumero*

Dipartimento di Economia e Produzione, Politecnico di Milano, P.za Leonardo da Vinci, 32, I20133 Milano, Italy

Received 1 May 1998; received in revised form 1 July 1999

Abstract

Despite nonmonotonic properties, weak convergence performance and possible erratic behavior, the standard subgradient optimization method is still one of the most widely adopted algorithm for solving the Lagrangean dual problem in practical applications. Several attempts have been made recently to improve the algorithm performance. In this paper we present a modified algorithm which employs a variable target value for the step length determination and considers a direction given by a conic combination of possibly all previously generated subgradients. Computational experience of the proposed algorithm on Traveling Salesman and Assignment problems of different sizes is reported.

Scope and purpose

In this paper, the problem of maximizing a concave and continuous function which is not differentiable everywhere is studied. Such a problem is quite common in the field of mathematical programming: for instance, whenever considering the Lagrangean relaxation of a linear programming problem, in order to solve the Lagrangean dual problem, one should optimize a dual objective function which is not differentiable everywhere. One of the most widely adopted solution method for solving the dual problem is subgradient optimization. Although many other procedures, with stronger convergence properties, have been developed (among the others, bundle methods or the Shor space dilation method), yet, subgradient optimization seems to have wide acceptability among researchers and remains as one of the most effective and useful technique for solving dual problems, especially for large problems with complex structures. In an attempt to improve the performance of the subgradient algorithm, researchers have tried to develop modifications and extensions of the basic scheme, aimed to reduce its main weaknesses (nonmonotonicity, weak convergence properties and lack of a definite termination criteria). In this paper, in order to further improve the computational efficiency of the procedure, we propose a modified subgradient algorithm which unifies some of the most promising results in the field of subgradient optimization. The provided solution scheme is applied to classical problems of the Operations Research area (Assignment and Traveling Salesman): the

* Fax: + 39-2-2399-2720.

E-mail address: francesca.fumero@polimi.it (F. Fumero)

PII: S0305-0548(99)00085-4

computational results obtained show a relevant improvement over similar algorithm previously proposed.

Keywords: Subgradient optimization; Nondifferentiable optimization; Lagrangean relaxation

1. Introduction

The problem of maximizing a concave and continuous function $L(u): \Re^n \to \Re$ which is not necessarily differentiable has been long studied: although the set of nondifferentiable points may have a null size, traditional techniques employed for smooth optimization cannot be adopted in this situation for two main reasons. First, the optimum is usually a point with no continuous derivative, hence using local approximations of the objective function may not be significant; second, the proposed optimality criteria and stop conditions (such as $\|\nabla f(x)\| < \varepsilon$) often do not make sense for nondifferentiable functions – consider for instance the minimization of the scalar function $f(x) = |x|$, where $\|\nabla f(x)\| = 1 \ \forall x \neq 0$. The problem of optimizing a nondifferentiable function is quite common in the field of mathematical programming: for instance, in the context of Lagrangean relaxation of discrete optimization problems [1–4]. Consider the following linear integer programming problem:

$$\begin{aligned} z = \min\ & cx \\ \text{s.t. } & Ax \leqslant b, \quad \text{(Pc)} \\ & Dx \leqslant d, \quad \text{(Ps)} \\ & x \geqslant 0 \text{ and integer,} \end{aligned} \tag{P}$$

where x is $n \times 1$, b is $m \times 1$, d is $k \times 1$, c is $1 \times n$, A is $m \times n$ and D is $k \times n$. Suppose that the constraints (Ps) are simple restrictions (such as network type), so that it would be much easier to solve the problem without the complicating constraints (Pc). Problem (P) may then be solved by pushing the complicating constraints (Pc) in the objective function via multipliers $u \geqslant 0$ and considering its Lagrangean relaxation:

$$\begin{aligned} z_{\mathrm{d}}(u) = \min\ & cx + u(Ax - b) \\ \text{s.t. } & Dx \leqslant d, \\ & x \geqslant 0 \text{ and integer.} \end{aligned} \tag{LS}$$

It is easy to verify that $z_{\mathrm{d}}(u) \leqslant z$ for all $u \geqslant 0$, so that it appears reasonable to define the dual problem (D) $z_{\mathrm{d}} = \max_{u \geqslant 0} z_{\mathrm{d}}(u)$, which provides a lower bound on the value of the objective function of problem (P). These bounds may be useful in pruning the branches in a branch-and-bound algorithm to find optimal solution for problem (P). Unfortunately, even when the integer constraints on variables x are removed, determining an optimal u to obtain z_{d} can be very time consuming. The dual function $z_{\mathrm{d}}(u)$ is a piecewise linear, concave and continuous function from $\Re^n$ to $\Re$ and it is nonsmooth whenever multiple solutions occur in the Lagrangean subproblem

(LS) [1], so that solving the dual problem involves an objective function which is not differentiable everywhere.

Iterative techniques have been developed to determine successive values of multipliers u using the subgradient of the objective function of problem (LS). Suppose x^k is an optimal solution of the Lagrangean subproblem (LS) at iteration k with $u = u^k$. The vector $(Ax^k - b)$ provides a subgradient g^k of $z_d(u)$ at point u^k: the next iterate (multipliers u^{k+1}) may then be determined according to the updating formula $u^{k+1} = u^k + t_k g^k$. This iterative method is commonly defined as subgradient optimization [5]. In Section 2, we provide a review of the literature on subgradient optimization methods, together with theoretical results, extensions and convergence conditions concerning this solution technique.

Notwithstanding the nonmonotonic properties and their possible erratic behavior, which in most cases causes slow convergence rates, subgradient optimization is one of the most popular solution method for solving the dual problem. Although many other techniques, with stronger convergence properties, have been developed (among the others, we may remember bundle methods [6–9] or the Shor space dilation method [10]), yet, subgradient optimization seems to have wide acceptability among researchers and continues to remain as one of the most effective and useful technique for solving dual problems, especially when large-scale applications are considered. This is mostly due to the simplicity of the algorithm's structure: more elaborate solution methods, with outstanding theoretical convergence performances, turn out to be much less competitive computationally, especially for large size problems with complex structure. Actually, to our knowledge, most of the applications described for these methods are limited to small-scale numerical examples, since the search for a descent direction is usually very time consuming. Consider, for instance, the necessity to handle a dilatation matrix in the Shor method or to solve, at each iteration, a quadratic problem in the bundle methods. More details on these issues may be found in [11].

For these reasons, many attempts have been made to improve the convergence properties of the subgradient algorithm and reduce its zigzagging behavior; so several extensions and heuristic modifications of the original scheme developed by Poljak [12] have been proposed. In this paper we develop a modified subgradient algorithm, which unifies some of the most promising results in the field of subgradient optimization. In Section 2, a brief survey of the alternative strategies developed to improve the performance of standard subgradient optimization is provided. The proposed algorithm is described in Section 3. The computational tests, conducted on the Lagrangean relaxation of Assignment problem and Traveling Salesman problem (TSP) of different sizes, are described in Section 4 and show a relevant improvement over similar algorithms previously proposed.

2. The subgradient optimization

Subgradient optimization can be considered as an extension to nondifferentiable functions of the gradient method for smooth optimization, in which information provided by the subgradient are incorporated: therefore, referring to the unconstrained problem $w = \max L(u)$ with $u \in \Re^m$, iterate u^{k+1} is generated according to the updating recursion $u^{k+1} = u^k + t_k g^k$, t_k being a scalar representing the step size and g^k a subgradient of the function $L(u)$ at the point u^k (i.e. $g^k \in \partial L(u^k)$, where $\partial L(u)$

denotes the subdifferential of $L(u)$ at point u). If we are considering a constrained maximization problem ($u \in U \subset \Re^n$), we should eventually project the obtained point onto the feasible region. It is worth noting that in the case of the Lagrangean dual, the solution of the Lagrangean subproblem naturally provides a subgradient of the dual function $L(u)$ at the point u^k, as shown in the example described in the introduction.

As a matter of fact, the obtained solution scheme is not an ascent algorithm, since the subgradient g^k does not necessarily provide an ascent direction. Yet, the method guarantees that the new point is closer, in the Euclidean sense, to the optimal point u^*, since, by definition, the vectors g^k and $(u^* - u^k)$ form an acute angle: a subgradient g^k at the point u^k is such that $L(u) - L(u^k) \leqslant (u - u^k)g^k$ for each u. Therefore, by selecting a sufficiently small step size $(0 < t_k < [2(L(u^*) - L(u^k))]/||g^k||^2)$, direction g^k allows to move towards the optimum, so that the Euclidean distance to the optimal solution is strictly decreasing ($||u^* - u^{k+1}|| < ||u^* - u^k||$). The choice of the step size is crucial for the convergence of the algorithm.

If the optimal value $w = L(u^*)$ happens to be known and $||g^k||$ is bounded, the suitable step size may be determined according to the formula

$$t_k = \frac{\lambda_k(w - L(u^k))}{||g^k||^2} \text{ with } \varepsilon_1 \leqslant \lambda_k \leqslant 2 - \varepsilon_2 \quad (\varepsilon_1, \varepsilon_2 > 0). \tag{1}$$

In this situation it is possible to assure geometric convergence to the optimal point [12]. Obviously, knowledge of the optimal value is in most cases not available. Anyway, similar results have been obtained when the unknown value is substituted by a lower/upper bound on it. For instance, in his seminal paper [12], Poljak has shown that if an underestimate $\underline{w}$ is used, the sequence $L(u^k)$ converges to $\underline{w}$ or a point u^k is obtained such that $L(u^k) \geqslant \underline{w}$.

In general, convergence to the optimal point is assured if one of the following conditions holds:

(a) $t_k \geqslant 0$, $\lim_{k\to\infty} t_k||g^k|| = 0$ and $\sum_{k=1}^{\infty} t_k||g^k|| = \infty$.

(b) $t_k \geqslant 0$, $\lim_{k\to\infty} t_k = 0$, $\sum_{k=1}^{\infty} t_k = \infty$ and $\{g^k\}$ is bounded for $k \geqslant 1$.

An interesting summary concerning the rates of convergence of the algorithm, for different choices of the step size and of the target value, is given in [13]. References to more recent results on this topic may be found in [11].

Most of the proposed applications, therefore, adopt formula (1), substituting to the unknown optimum a lower/upper bound on it and assuring previous convergence conditions [3, 5]. Computational tests seem to indicate that in many cases the requirement of series divergence is not essential in practice for the convergence of the algorithm [5].

Despite the great simplicity of the method, its nonmonotonicity, its weak convergence properties and the lack of a definite termination condition have promoted in recent years the development of more efficient variants of this basic scheme. Most of the proposed modifications, aimed primarily at accelerating the method, have been developed along two main research streams: one related to the determination of a more accurate step size and the other related to the search direction definition.

For the first stream, convergence conditions for the choice of step size have been extended and generalized [14] and some attempts have been made to simplify the projection step for constrained problems [15]. Yet, the best results seem to be obtained by the so-called Variable Target Value

techniques [16], where the step size is defined by substituting to the optimum value w used in formula (1) a suitable estimation $\tilde{w}_k$, which is iteratively updated. In the simplest case, referring to the Lagrangean dual, an upper bound $\tilde{w}_k$ may be determined, at each iteration k of the algorithm, by an heuristic procedure which derives a primal feasible solution from the solution to the Lagrangean problem associated to current multipliers. An interesting application of this solution scheme is described in [17], which considers the problem of Local Area Network (LAN) connection. In general, the estimate $\tilde{w}_k$ may be obtained considering a nonincreasing sequence $\{U_k\}$ (upper bounds to the optimal value w) and a nondecreasing sequence $\{L_k\}$ (lower bounds to w), such that $L_k \geqslant L(u^k)$, as follows:

$$\tilde{w}_k = \alpha_k U_k + (1 - \alpha_k)L_k \quad \text{with } 0 \leqslant \alpha_k \leqslant 1.$$

Under suitable conditions [16], the algorithm converges to the optimum without requiring prior knowledge of the optimal objective value and it implicitly provides a termination criterion in the form $U_k - L_k \leqslant \varepsilon$ (for ε sufficiently small).

The second main research stream is motivated by the attempt to control the erratic behavior of the algorithm. Since the subgradient method is not an ascent algorithm, it may define an oscillating path (zigzagging phenomena) towards the optimal point, slowing down the search of the optimum. For instance, in the case of piecewise linear functions, the algorithm may iterate among two polyhedral regions Y_j and Y_h with different optimal solutions, as shown in Fig. 1. A detailed description of the phenomenon is provided in [18], while a simple example of this situation may be found in [19].

The natural observation that the Markov nature of the subgradient method, which does not preserve memory of previous steps, is the chief cause of this zigzagging behavior and of the slow convergence rate naturally leads to the idea of employing both subgradients at the current point and at previous iterations. Hence, modified subgradient techniques have been proposed [20–23]: the search direction d^k is defined considering a suitable combination of the current and previous subgradients ($d^k = \mu_k g^k + v_k d^{k-1}$) and iterate $k+1$ is obtained as usual ($u^{k+1} = u^k + t_k d^k$). A special case occurs if the direction d^k is an ε_k-subgradient of the function $L(u)$ in u^k, where an ε-subgradient $\bar{g}'$ at point $\bar{u}$ is such that $L(u) - L(\bar{u}) \leqslant (u - \bar{u})\bar{g}' + \varepsilon$, for each u. For instance, the search direction is an ε-subgradient whenever a convex combination of generated subgradients is used, as shown in [24]. In this special case, we may then derive the following conditions for

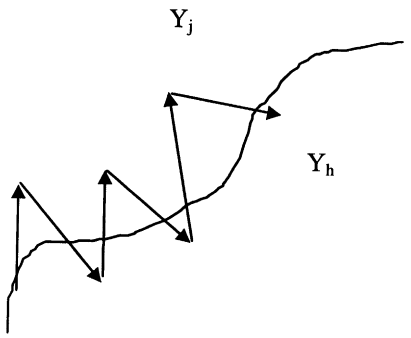

Fig. 1. The subgradient alternating components.

convergence to the optimum:

$$t_k \geqslant 0, \quad \lim_{k\to\infty} t_k\|d^k\| = 0, \quad \sum_{k=1}^{\infty} t_k\|d^k\| = \infty \quad \text{and} \quad \lim_{k\to\infty} \varepsilon_k = 0.$$

In general, if the multipliers updating formula is in the form $u^{k+1} = u^k + \sum_{i\in I_k} s_k^i g^i$, where I_k is a subset of the considered indexes $\{1,\ldots,k\}$ and s_k^i are suitable nonnegative weights, the modified subgradient scheme converges to the optimum if [24]

$$\lim_{k\to\infty} \left\|\sum_{i\in I_k} s_k^i g^i\right\| = 0, \quad \sum_{k=1}^{\infty} \left\|\sum_{i\in I_k} s_k^i g^i\right\| = \infty \quad \text{and} \quad \lim_{k\to\infty} \varepsilon_k = 0,$$

where $\varepsilon_k = \sum_{i\in I_k}(s_k^i/s_k)\varepsilon_k^i$, $s_k = \sum_{i\in I_k} s_k^i$ and $\varepsilon_k^i = L(u^k) - L(u^i) - g^i(u^k - u^i)$.

Many authors (among the others [22,25–27]) have shown that the modified method has stronger convergence properties and is superior to the standard subgradient method both theoretically and computationally. In [24] an in-depth analysis of convergence properties of these subgradient schemes and a wide survey of applications are presented.

3. A modified subgradient algorithm with variable target value

In this section a new subgradient optimization algorithm is proposed with the aim of improving the computational efficiency by means of both the research streams described in the previous section. Therefore, in the solution scheme a variable target value is used in the step size definition formula and, in the multipliers updating rule, a modification to the direction defined by the subgradient is obtained on the basis of previous iterations of the algorithm. The algorithm consists of two phases: in both of them convergence is accelerated by a suitable step size definition and by using an improved search direction; moreover, in the second phase, some of the parameters are fixed with the aim of accelerating convergence to the optimum. A detailed description of the algorithm may be found in the second part of this section and in the appendix. In the following, we first provide the basic idea of the algorithmic scheme.

For what concerns the target value, we follow an approach similar to the one suggested in [28]. As the authors show, the optimal value for the step size should be obtained by formula

$$t_k^* = \frac{g^k(u^* - u^k)}{\|g^k\|^2} \geqslant \frac{w - L(u^k)}{\|g^k\|^2}. \tag{2}$$

When the optimal multipliers or the optimal objective value are not known a priori, an estimate $\bar{L}$ can be used to substitute w:

$$t_k = \frac{\bar{L} - L(u^k)}{\|g^k\|^2}. \tag{3}$$

The authors suggest to periodically update the estimate $\bar{L}$ so that the value $(\bar{L} - w)/\|g_k\|^2$ results to be directly related to the gap in the inequality of (2). Therefore, a variable target value algorithm

is obtained, in which the estimate of the optimal value is determined as $\bar{L} = \alpha_r L^0 + (1 - \alpha_r)L^c$, where L^0 is a fixed upper bound to the unknown optimum w, L^c the best objective value so far known, and $\{\alpha_r\}$ a monotonic decreasing sequence such that $\alpha_0 = 1$ and $\alpha_r \to \varepsilon_0 > 0$. A reasonable strategy will associate a significant weight to the fixed upper bound in the first steps of the algorithm and then let the weight decrease more rapidly later as the current best L^c increases and gains significance. According to [28], we adopt therefore the following updating scheme for function α_r, which is also represented in Fig. 2:

$$\alpha_r = \begin{cases} \varepsilon_0 & \text{if } r \geqslant r_2; \\ e^{-0.6933(r/r_1)^{3.26}} & \text{otherwise.} \end{cases}$$

As shown in Fig. 2, parameter r_1 identifies the point at which $\alpha_{r_1} = 1/2$, so that the upper bound L^0 and the current best L^c equally concur in determining the target value. Therefore, parameter r_1 provides control over the procedure, allowing to delay or anticipate the shift of the dominating weight from L^0 to L^c. Such a choice may for instance depend on the significance of the available upper bound L^0. On the other side, parameter r_2 allows to define when the procedure should pass onto the second phase, in which α_r is frozen at value ε_0 (see Fig. 2). Once parameter ε_0 is chosen, r_2 is defined as the smallest integer value which satisfies $\varepsilon_0 \geqslant e^{-0.6933(r_2/r_1)^{3.26}}$. Suitable values for parameters r_1 and ε_0 should be chosen according to the specific problem considered: as a general intuitive criteria, one should prefer relatively higher values for r_1 and lower values for ε_0 whenever L^0 is close to w and vice versa otherwise.

The main idea of the algorithm is that, if the objective function does not show any improvement over a certain number of iterations, we are probably using a too large step: we therefore increase the value of r, so that α_r and $\bar{L}$ decrease and we may obtain, by formula (3), a smaller step size. The introduction of a lower value ε_0, which is employed in the second phase of the algorithm, allows to support convergence to the optimum and to improve the performance of the algorithm. Moreover, differently from [28], a coefficient $1/\beta_r$ has been introduced in the step size definition formula (3): In analogy to the criteria usually suggested to update parameter λ_k in formula (1), β_r is increased by

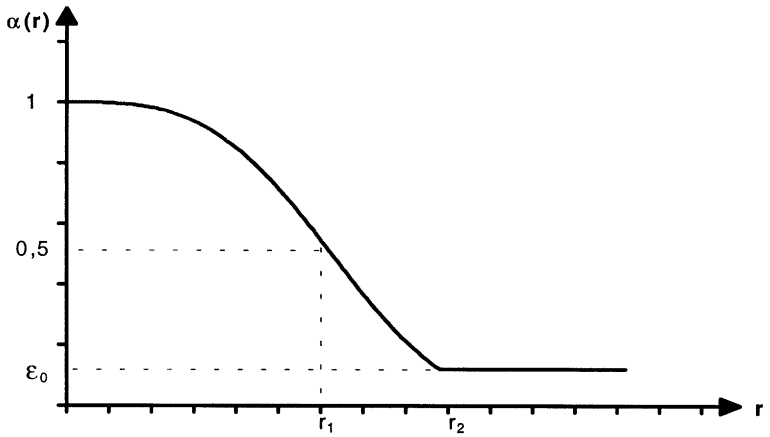

Fig. 2. Function α.

two if there is no improvement in the objective function, in an attempt to obtain a further reduction of the step size. In phase II, to further accelerate the method, the strategy is strengthened: β_r is then doubled whenever the algorithm does not seem to get any significant improvement.

Furthermore, with the aim of improving the performance of the algorithm, we resort to modified techniques, trying to employ, instead of the subgradient, a more efficient search direction: therefore, whenever the current subgradient and the direction obtained at the previous iteration form an obtuse angle, we operate a correction of the search direction. As it is well known, this may assure an improvement in the decrease of the Euclidean distance from the optimum. Therefore the search direction d^k is determined, according to a similar scheme proposed in [20], as

$$d^k = g^k + \xi_k d^{k-1}, \tag{4}$$

with

$$\xi_k = \begin{cases} -\gamma \dfrac{d^{k-1} g^k}{\|d^{k-1}\|^2} & \text{if } d^{k-1} g^k \leqslant 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

This weighting scheme is strictly related to similar approaches proposed for the conjugate gradient method for smooth optimization.

Therefore, it is possible to provide the following description of the modified subgradient algorithm. At a generic iteration k, we consider the subgradient g^k of L in u^k: using formula (4), we obtain d^k, by fixing ξ_k according to (5); we compute the step size $t_k = (1/\beta_r)[(\bar{L} - L(u^k))/\|d^k\|^2]$ (with $\bar{L} = \alpha_r L^0 + (1 - \alpha_r)L^c$ and $\alpha_r = e^{-0.6933(r/r_1)^{3.26}}$) and we determine then the next iterate $u^{k+1} = u^k + t_k d^k$ (with the use of a projection operation, if necessary). Such iteration is considered a failure if $L(u^{k+1}) \leqslant L^c + \varepsilon$, ε being a prespecified positive tolerance level. For every $\bar{v}_1$ consecutive failures, we increase r by one, updating also the values of α_r and β_r, and we assume as starting point for the next iteration the solution u^c associated with the best obtained value L^c. As noted in [28], this resetting operation avoids straying of the sequence for too many iterations without improving. As long as $r < r_2$, we repeat a similar iteration.

When r reaches the value r_2 (see Fig. 2), we enter the second phase of the algorithm. The updating scheme is similar to the one adopted in phase I, with some minor modifications concerning the setting of some parameters. Firstly, we fix $\alpha_r = \varepsilon_0$, updating the estimate of the optimum according to $\bar{L} = \varepsilon_0 L^0 + (1 - \varepsilon_0)L^c$. It is worth noting that, in contrast to similar previously proposed modified algorithms [20, 27], the target value in the step size definition is still iteratively updated, even if parameter α_r is now fixed to the value ε_0: every improvement of the current best value L^c implies a change in the target value $\bar{L}$. Moreover, with respect to phase I, we change the updating criteria for β_r, which is now doubled for every $\bar{v}_2$ consecutive failures of the algorithm; in this case, the resetting to the best known solution is performed only if β_r is less than some fixed integer value $\bar{\beta}$: beyond that value, the step size is usually very small, so that the current point is not far from the point associated with the best known objective value and it is not worth changing the starting point in the next iteration. If the objective function increases in less than $\bar{v}_2$ iterations, we update the value L^c and we halve parameter β_r in order to avoid an excessive smoothing of the search process due to a very small step size. The termination criteria considered

are the following:

1. $\|g^k\| = 0$;
2. $t_k\|g^k\| \leqslant LIM1$ in phase I or $t_k\|d^k\| \leqslant LIM\,2$ in phase II for a fixed number (MAX) of iterations;
3. maximum number of iterations (KMAX $= \bar{k}$) reached.

A detailed description of the iterative procedure in the format of an algorithmic scheme may be found in the Appendix. In the following section, we analyze the computational performances of the algorithm, providing results obtained on TSP and Assignment problems.

4. Computational results

The proposed algorithm has been applied to two standard integer linear programming problems of the Operations Research area: the symmetric TSP and the Assignment problem. In the following, we briefly provide, for each problem, a possible formulation, the Lagrangean relaxation, and the expression of a corresponding subgradient. More details on these problems may be found in [28].

Letting X be the set of all 1-trees of a graph [29] and c_{ij} the cost of link (i,j), the symmetric traveling salesman problem can be stated as follows:

$$\begin{aligned} &\min \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} c_{ij}x_{ij} \\ &\text{s.t.} \sum_{\substack{j=1 \\ j\neq i}}^{m} x_{ij} + \sum_{\substack{j=1 \\ j\neq i}}^{m} x_{ji} = 2 \qquad \text{for } i = 1,\ldots,m,\ x \in X \end{aligned} \tag{6}$$

Incorporating the explicit restrictions (6) into the objective function via a suitable vector $u \in \Re^m$ of Lagrangean multipliers, one obtains the following dual function, which has to be maximized:

$$L(u) = -2\sum_{i=1}^{m} u_i + \min\left\{\sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} (c_{ij} + u_i + u_j)x_{ij} : x \in X\right\}.$$

It may easily be verified that, given a vector $\bar{u}$, if $\bar{x}$ optimizes $L(\bar{u})$, then a vector $\bar{g}$ whose ith component is $\bar{g}_i = (\sum_{j=1,j\neq i}^{m} \bar{x}_{ij} + \sum_{j=1,j\neq i}^{m} \bar{x}_{ji} - 2)$, is a subgradient of $L(u)$ at $\bar{u}$.

If c_{ij} is the cost of assigning i to j, the linear assignment problem can be formulated as

$$\begin{aligned} &\min \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}x_{ij} \\ &\text{s.t.} \sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1,\ldots,m, \\ &\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1,\ldots,m,\ x_{ij} \geqslant 0,\ i,\ j = 1,\ldots,m. \end{aligned} \tag{7}$$

By pricing each constraint (7) via a Lagrangean multiplier u_i, a dual problem in the form $\max L(u)$ is generated, where

$$L(u) = \min \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{m} u_i\left(\sum_{j=1}^{m} x_{ij} - 1\right)$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, m,\ x_{ij} \geqslant 0,\ i, j = 1, \ldots, m.$$

Moreover, if $\bar{x}$ determines $L(\bar{u})$, a subgradient of $L(u)$ at $\bar{u}$ is obtained by considering a vector $\bar{g}$ with components $g_i = (\sum_{i=1}^{m} \bar{x}_{ij} - 1)(i = 1, \ldots, m)$.

The selected instances of the two problems considered are similar to those used in [28] and represent networks of different sizes, varying from 10 to 57 nodes: moreover, few larger problems, with 100 nodes, have been considered to further evaluate the proposed algorithm. Many of the numerical examples selected refer to data from the published literature: in particular, the data for the 10-nodes problem is from [30], those for the 25- and 48-nodes problems are from [29], the 33- and 57-nodes problems are from [31] and the 42-nodes problem is from [32]. The 100-nodes problem is available on the OR-library on Internet [33]. Finally, five random instances, describing problems with 100 nodes, have been generated.

Tables 1–7 report the computational results obtained for the different instances of the Assignment (ASS) and the TSP problems reported in the literature. In each table, the following

Table 1
Computational results for the 10-nodes assignment and traveling salesman problems

10-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	326	326	326	378	378	378
Upper bound	362	362	362	411	411	411
Best	326	326	326	378	378	378
It. Best	11	35	23	28	32	19
Stop	1	1	1	2	3	3
It. Stop	11	35	23	67	200	200
Value of switch	—	—	—	378	378	—
It. Switch	—	—	—	44	39	—
It.10	323.515	309.8	284.352	371.739	371.739	373.593
It.20	—	321.4	323.083	376.45	376.45	378
It.30	—	323.5	—	378	377.397	378
It.40	—	—	—	378	378	378
It.50	—	—	—	378	378	378
It.75	—	—	—	—	378	378
It.100	—	—	—	—	378	378
It.125	—	—	—	—	378	378
It.150	—	—	—	—	378	378
It.175	—	—	—	—	378	378

Table 2
Computational results for the 25-nodes assignment and traveling salesman problems

25-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	1281	1281	1281	1711	1711	1711
Upper bound	1470	1470	1470	2037	2037	2037
Best	1281	1280.97	1281	1711	1711	1711
It. best	82	152	122	52	56	42
Stop	2	3	3	2	3	3
It. stop	113	200	200	140	200	200
Value of switch	1280.99	1277.9	—	1711	1711	—
It. Switch	80	46	—	93	56	—
It.10	1232.53	1158.7	1016	1627.18	1627.18	1595.69
It.20	1264.33	1191.2	1016	1677.88	1677.88	1672.46
It.30	1273.49	1241.3	1211.35	1699.53	1693.89	1699.89
It.40	1279.91	1271.8	1256.56	1702.48	1703.84	1708.38
It.50	1280.77	1279.9	1276.45	1710.58	1710.66	1711
It.75	1280.99	1280.8	1280.31	1711	1711	1711
It.100	1281	1280.9	1280.96	1711	1711	1711
It.125	—	1280.96	1281	1711	1711	1711
It.150	—	1280.96	1281	—	1711	1711
It.175	—	1280.96	1281	—	1711	1711

information are provided:

- known optimum value;
- starting upper bound employed;
- best value obtained;
- iteration at which the best value is obtained;
- stop condition (the number refers to the list in the previous section);
- iteration at which the algorithm ends;
- Lagrangean value when switching between Phase I and II takes place;
- iteration at which switching takes place;
- best values obtained at different iterations of the algorithm.

Results obtained by applying the variable target value modified algorithm developed in the previous section have been compared to those reported by [28], which employed a similar variable target scheme. In particular, the proposed algorithm adopts the same updating formula for function α_r, although some parameters of the algorithm are differently controlled in our case, as pointed out in Section 3. Yet, the main difference among the two algorithms lies in the employment of the modified direction: in [28] a simple subgradient is used to identify the search direction of the algorithm. Moreover, we also tested a subgradient procedure similar to the original proposal of Held et al. [5]: the step size is defined according to formula (1), by initially setting λ_0 to value 2 for m iterations, m being a suitable parameter depending on the size of the problem; then, both the

Table 3
Computational results for the 33-nodes assignment and traveling salesman problems

33-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	9948	9948	9948	10 861	10 861	10 861
Upper bound	11 219	11 219	11 219	11 994	11 994	11 994
Best	9948	9947.9	9948	10 861	10 850.1	10 861
It. Best	100	194	136	53	194	43
Stop	2	3	3	2	3	3
It. Stop	133	200	200	120	200	200
Value of switch	9948	9903.7	—	10 861	10 842.8	—
It. Switch	105	44	—	74	47	—
It.10	9169.32	8590	8590	10 732.6	10 732.6	10 621
It.20	9719.4	9251	8590	10 804.7	10 804.7	10 715.9
It.30	9794.99	9585.9	8590	10 819.8	10 815.7	10 793
It.40	9916.61	9903.7	9050	10 839.6	10 834.2	10 851.5
It.50	9930.23	9931.3	9466.84	10 856.6	10 845.7	10 861
It.75	9946.39	9946	9938.9	10 861	10 849.5	10 861
It.100	9948	9947	9947.64	10 861	10 849.8	10 861
It.125	9948	9947.5	9947.96	—	10 850	10 861
It.150	—	9947.8	9948	—	10 850.1	10 861
It.175	—	9947.8	9948	—	10 850.1	10 861

Table 4
Computational results for the 42-nodes assignment and traveling salesman problems

42-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	532	532	532	699	699	699
Upper bound	581	581	581	969	969	969
Best	532	531.99	532	696.98	688.449	696.999
It. best	116	163	165	197	198	141
Stop	2	3	3	3	3	3
It. stop	131	200	200	200	200	200
Value of switch	531.999	530.9	—		687.249	—
It. Switch	104	62	—	Always I	46	—
It.10	491.447	487	454	666.599	663.024	631.142
It.20	524.161	496.4	454	676.267	674.471	655.438
It.30	526.423	504.8	454	679.982	683.812	655.438
It.40	530.469	520.2	454	690.25	686.046	655.438
It.50	530.798	528.3	508.671	693.079	687.743	681.764
It.75	531.876	531.6	526.226	694.634	688.191	693.74
It.100	531.999	531.9	531.797	695.784	688.323	696.848
It.125	532	531.96	531.987	696.177	688.401	696.992
It.150	—	531.98	531.999	696.545	688.42	696.999
It.175	—	531.98	532	696.935	688.42	696.999

Table 5
Computational results for the 48-nodes assignment and traveling salesman problems

48-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	9870	9870	9870	11 445	11 445	11 445
Upper bound	14 072	14 072	14 072	14 241	14 241	14 241
Best	9870	9869.2	9869.99	11 441.6	11 434.3	11 442
It. best	131	198	176	200	171	176
Stop	2	3	3	3	3	3
It. stop	162	200	200	200	200	200
Value of switch	9869.99	9771.9	—	11 441.5	11 428.8	—
It. Switch	124	47	—	154	45	—
It.10	9220.8	8827.5	8757	11 197.9	11 197.9	11 048.7
It.20	9513.51	8827.5	8757	11 322.5	11 327.7	11 151.5
It.30	9686.97	9378.6	8757	11 357.4	11 386.4	11 221
It.40	9796.32	9731.7	8757	11 419.3	11 422.1	11 221
It.50	9839.66	9844.3	8757	11 426.1	11 432.4	11 320
It.75	9868.83	9860.3	8757	11 434.6	11 433.9	11 320
It.100	9869.91	9864.6	9789.73	11 439	11 434.2	11 440
It.125	9869.99	9867.8	9865.17	11 440.7	11 434.2	11 441.7
It.150	9870	9868.4	9869.77	11 441.5	11 434.3	11 441.9
It.175	—	9869.2	9869.98	11 441.5	11 434.3	11 442

Table 6
Computational results for the 57-nodes assignment and traveling salesman problems

57-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	10 553	10 553	10 553	12 908	12 908	12 908
Upper bound	12 311	12 311	12 311	15 889	15 889	15 889
Best	10 553	10 552.6	10 553	12 907.5	12 891.3	12 907.5
It. Best	90	189	174	197	199	186
Stop	2	3	3	3	3	3
It. Stop	130	200	200	200	200	200
Value of switch	10 553	10 491.4	—	12 906.7	12 879.1	—
It. Switch	105	51	—	121	41	—
It.10	10 265.1	9263.8	8928	12 630.3	12 630.3	12 130.6
It.20	10 362.6	9519.2	8928	12 735.9	12 701.4	12 282.3
It.30	10 456.4	10 108.2	8928	12 760.8	12 835.1	12 361.3
It.40	10 515.9	10 423.8	8928	1 2851.4	12 874.9	12 423.2
It.50	10 547.4	10 491.4	8928	12 869.3	12 886.9	12 423.2
It.75	10 552.8	10 533.6	9777.28	12 902	12 889.6	12 735.6
It.100	10 553	10 543.4	10 327.7	12 904.5	12 890.3	12 877.3
It.125	10 553	10 548.2	10 546.2	12 906.9	12 890.7	12 905.9
It.150	—	10 550.7	10 552.2	12 907.2	12 890.9	12 907.4
It.175	—	10 552.1	10 553	12 907.4	12 891.1	12 907.5

Table 7
Computational results for the 100-nodes assignment and traveling salesman problems

100-nodes problems	FF	BS	HWC	FF	BS	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Optimum	305	305	305	384	384	384
Upper bound	420	420	420	670	670	670
Best	304.939	304.205	303.924	380.393	376.129	374.902
It. Best	200	138	200	95	114	189
Stop	3	3	3	2	3	3
It. Stop	200	200	200	131	200	100
Value of switch	304.907	303.68	—			—
It. Switch	163	68	—			—
It.10	284.316	285.459	245	323.219	323.219	274
It.20	300.216	286.817	245	355.306	335.483	274
It.30	302.016	294.441	245	356.993	375.802	274
It.40	303.228	301.951	245	365.633	375.802	274
It.50	303.389	302.917	245	366.206	375.952	274
It.75	304.373	304.085	245	380.36	375.952	274
It.100	304.558	304.108	245	380.393	375.952	274
It.125	304.833	304.198	287.781	380.393	376.129	305.825
It.150	304.891	304.205	288.455	—	376.129	309.209
It.175	304.92	304.205	298.663	—	376.129	340.424

value of λ_k and the number of iterations are successively halved until a threshold value is reached. No attempt is made to use variable target values or modified directions.

For the two problems analyzed, the first column of the tables refers to results obtained by applying the algorithm proposed in this paper (FF), the second column (BS) reports results provided by [28], while the last one (HWC) concerns the solution scheme inspired by [5].

All the computational tests were executed on a Pentium 133 using C++ language and no significant differences among the three algorithms emerged in computational times. As can be noticed, in the case of the assignment problems the proposed algorithm outperforms the other solution schemes, both for the bound obtained and for the intermediate values provided: in particular, the algorithm seems to be fairly rapid in improving the objective function value in the initial stages. For what concerns the traveling salesman problems, similar remarks hold: yet, with the exception of the 100-nodes problem, bounds provided by the HWC scheme are slightly better then those obtained with the proposed algorithm. This seems to be caused by a significant acceleration in the last iterations of the procedure, since the (FF) scheme generally provides higher objective values in the initial steps.

Tables 8–12, which are similar to the previous ones, illustrate the main results of the computational experiences conducted on the 100-nodes problems randomly generated. In this case also it is possible to verify that, with the exception of problem 110-d (Table 11), the proposed algorithm provides higher lower bounds and significantly outperforms the alternative solution strategies in the first steps of the implementation.

Table 8
Computational results for the 100-nodes assignment and traveling salesman problems (random stream a)

100 (a) nodes	FF	B&S	HWC	FF	B&S	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Best	300.983	300.62	300.755	379.04	353.651	365.411
It. best	153	92	200	78	86	193
It. stop	172	200	200	78	200	200
It.10	295.175	294.77	269.335	344.409	344.409	336.136
It.20	299.537	296.552	275.242	356.854	344.409	336.756
It.30	299.979	297.505	276.127	367.783	352.955	348.572
It.40	300.121	299.126	278.578	376.82	353.632	348.572
It.50	300.4	300.251	281.121	376.82	353.632	348.572
It.75	300.729	300.607	281.121	379.339	353.632	355.673
It.100	300.887	300.62	281.121	—	353.651	355.673
It.125	300.954	300.62	296.239	—	353.651	356.007
It.150	300.981	300.62	297.117	—	353.651	356.007
It.175	—	300.62	299.138	—	353.651	359.483

Table 9
Computational results for the 100-nodes assignment and traveling salesman problems (random stream b)

100 (b) nodes	FF	B&S	HWC	FF	B&S	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Best	197.862	197.463	197.828	265.41	251.559	258.003
It. best	130	108	200	52	46	162
It. stop	141	200	200	79	200	200
It.10	195.057	192.921	187.967	241.709	234.435	231.66
It.20	195.912	196.04	190.381	251.667	243.379	250.8
It.30	197.22	196.04	190.467	264.244	251.247	250.8
It.40	197.381	196.814	190.467	265.399	251.532	250.91
It.50	197.494	197.008	190.467	265.399	251.559	257.973
It.75	197.688	197.46	190.798	265.41	251.559	257.973
It.100	197.802	197.462	190.798	—	251.559	257.973
It.125	197.854	197.463	196.312	—	251.559	257.973
It.150	—	197.463	196.312	—	251.559	257.973
It.175	—	197.463	197.346	—	251.559	258.003

Therefore, computational results obtained seem really encouraging, above all in the context of large-scale problems, for which an acceleration in the first steps of the algorithm may be really valuable. As a matter of fact, very often, in large applications especially, computational times are crucial and it is not possible to run the subgradient procedure until the optimal solution is found. In this sense, it is particularly relevant to note that the advantage of the algorithm proposed grows

Table 10
Computational results for the 100-nodes assignment and traveling salesman problems (random stream c)

100 (c) nodes	FF	B&S	HWC	FF	B&S	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Best	334.996	334.9	334.75	463.402	457.975	444.97
It. best	89	120	200	81	149	188
It. stop	107	200	200	111	200	200
It.10	315.176	318.681	267.062	402.077	412.885	358.596
It.20	331.337	322.219	275.261	433.189	424.232	365.436
It.30	333.632	324.357	275.261	454.828	450.131	365.436
It.40	334.361	329.965	275.261	454.828	455.342	384.886
It.50	334.858	333.786	275.261	461.406	455.929	384.886
It.75	334.979	334.87	275.261	463.399	457.921	384.886
It.100	334.996	334.892	275.261	463.402	457.921	384.886
It.125	—	334.9	323.064	—	457.921	407.768
It.150	—	334.9	324.057	—	457.975	407.768
It.175	—	334.9	332.589	—	457.975	415.133

Table 11
Computational results for the 100-nodes assignment and traveling salesman problems (random stream d)

100 (d) nodes	FF	B&S	HWC	FF	B&S	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Best	3258.83	3275.57	3341.63	3736.02	3706.91	3775.57
It. best	20	94	200	90	99	198
It. stop	55	200	200	132	200	200
It.10	3258.67	3274.88	2895.2	3538.96	3538.96	3585.84
It.20	3258.83	3274.88	1895.2	3584.66	3559.63	3589.93
It.30	3258.83	3275.06	3095.48	3684.72	3689.24	3615.39
It.40	3258.83	3275.12	3070.42	3722.01	3705.92	3692.52
It.50	3258.83	3275.12	3070.42	3735.48	3706.81	3714.67
It.75	—	3275.49	3070.42	3735.68	3706.81	3714.67
It.100	—	3275.57	3070.42	3736.02	3706.91	3714.67
It.125	—	3275.57	3208.69	3736.02	3706.91	3714.67
It.150	—	3275.57	3286.7	—	3706.91	3744.14
It.175	—	3275.57	3331.34	—	3706.91	3764.48

with the size of the instances considered (see for instance Table 7). Moreover, the importance of obtaining significant bounds already in the first iterations of the algorithm may be further appreciated considering the use of Lagrangean relaxation for fathoming nodes in a Branch-and-Bound tree. The main weakness of the algorithm, emerging from computational tests implemented, concerns the fine tuning of the parameters: in particular, it is critical suitable setting of parameters r_2, ε_0, $\bar{v}_1$ and $\bar{v}_2$. Unfortunately, the "optimal" setting of the parameters seems to be strictly

Table 12
Computational results for the 100-nodes assignment and traveling salesman problems (random stream e)

100 (e) nodes	FF	B&S	HWC	FF	B&S	HWC
Problem	ASS	ASS	ASS	TSP	TSP	TSP
Best	1510	1509.87	1509.23	1695.84	1685.2	1681.23
It. best	124	165	200	86	49	189
It. stop	134	200	200	116	200	200
It.10	1471.51	1475.25	1369	1590.37	1590.37	1545.93
It.20	1500.25	1479.46	1369	1641.51	1609.93	1545.93
It.30	1500.25	1490.42	1369	1667.83	1668.9	1545.93
It.40	1507.29	1497.4	1369	1681.91	1676.61	1545.93
It.50	1509.22	1506.52	1369	1683.74	1685.2	1545.93
It.75	1509.9	1509.21	1369	1695.83	1685.2	1545.93
It.100	1509.98	1509.82	1369	1695.84	1685.2	1545.93
It.125	1510	1509.85	1480.37	—	1685.2	1629.3
It.150	—	1509.85	1486.73	—	1685.2	1629.3
It.175	—	1509.87	1496.1	—	1685.2	1664.89

Table 13
Values of the parameters adopted in the computational tests

	r_2	ε_0	$\bar{v}_1 = \bar{v}_2$
Assignment	4–5	0.001	4
Travelling Salesman	3–5	0.01	5

problem dependent: yet, a similar remark also holds for the algorithm proposed in [28], while application of the original scheme (HWC) is generally straightforward (except for selecting parameter m). In order to provide interested readers with some useful hints, we reported in Table 13 the values assigned to the parameters in the two problems considered in the computational tests.

Actually the developed algorithm has also been tested on larger problems, with interesting results: in particular, the proposed scheme has been employed to solve several optimization models concerning real applications in the context of logistic and production planning models [34–36].

Appendix

We provide in the following a detailed description of the iterative procedure in the format of an algorithmic scheme. All the notations adopted correspond to definitions provided in the paper, with the exception of symbol P_U: this denotes the projection operator which may be necessary to generate points $u \in U$ in constrained problems.

Initialization

Choose a starting vector u^1 and compute $L(u^1)$ and $g^1 \in \partial L(u^1)$. If $\|g^1\| = 0$, STOP;
Set $r = 0$, $v = 0$, $z = 0$, $k = 1$, $\alpha_0 = 1$, $\beta_r = 1$ and $\bar{L} = L^0 \geqslant w^*$.
Set $d^1 = g^1$ and $(u^c, L(u^c), d^c) = (u^1, L(u^1), d^1)$.

Test

If $r < r_2$, $\rightarrow$ *Phase I*; otherwise $\rightarrow$ *Phase II.*

Phase I

1. Compute $t_k = (1/\beta_r)[(\bar{L} - L(u^k))/\|d^k\|^2]$;
2. If $t_k\|d^k\| >$ LIM1, set $z = 0$. Otherwise, $z = z + 1$ and if $z = MAX$, STOP;
3. Determine $u^{k+1} = P_U(u^k + t_k d^k)$, $L(u^{k+1})$ and $g^{k+1} \in \partial L(u^{k+1})$. If $\|g^{k+1}\| = 0$, STOP;
4. $k = k + 1$. If $k = \bar{k}$, STOP.
5. If $d^{k-1}g^k < 0$, set $\xi_k = -\gamma(d^{k-1}g^k/\|d^{k-1}\|^2)$ and $d^k = g^k + \xi_k d^{k-1}$; otherwise $d^k = g^k$.
6. If $L(u^k) \geqslant L^c + \varepsilon$, $\rightarrow$ *Phase I*: *Step 8*;
7. $v = v + 1$
   If $v < \bar{v}_1$, $\rightarrow$ *Phase I*: *Step 1.*;
   otherwise – $v = 0$, $r = r + 1$;
   – $\beta_r = \beta_{r-1} + 2$;
   – $\alpha_r = e^{-0.6933(r/r_1)^{3.26}}$;
   – $\bar{L} = \alpha_r L^0 + (1 - \alpha_r)L^c$;
   – $(u^k, L(u^k), d^k) = (u^c, L(u^c), d^c)$;
   – $\rightarrow$ *Test*;
8. Set – $(u^c, L(u^c), d^c) = (u^k, L(u^k), d^k)$
   – $\bar{L} = \alpha_r L^0 + (1 - \alpha_r)L^c$
   – $v = 0$, and $\rightarrow$ *Phase I*: *Step 1.*;

Phase II

1. Set $\beta_k = \beta_r$, $z = 0$;
2. Compute $t_k = (1/\beta_k)[(\bar{L} - L(u^k))/\|d^k\|^2]$;
3. If $t_k\|d^k\| >$ LIM2, set $z = 0$; otherwise, $z = z + 1$ and if $z = MAX$, STOP;
4. Determine $u^{k+1} = P_U(u^k + t_k d^k)$, $L(u^{k+1})$ and $g^{k+1} \in \partial L(u^{k+1})$. If $\|g^{k+1}\| = 0$, STOP;
5. $k = k + 1$. If $k = \bar{k}$, STOP;
6. If $d^{k-1}g^k < 0$, set $\xi_k = -\gamma(d^{k-1}g^k/\|d^{k-1}\|^2)$ and $d^k = g^k + \xi_k d^{k-1}$; otherwise $d^k = g^k$.
7. If $L(u^k) \leqslant L^c + \varepsilon$, $\rightarrow$ *Phase II*: *Step 8*; otherwise $\rightarrow$ *Phase II*: *Step 9;*
8. $(u^c, L(u^c), d^c) = (u^k, L(u^k), d^k)$, $\bar{L} = \alpha_r L^0 + (1 - \alpha_r)L^c$ and $\beta_k = \beta_{k-1}/2$;
9. $v = v + 1$.
   If $v < \bar{v}_2$: $\rightarrow$ *Phase II*: *Step 2*;
   Otherwise: – $v = 0$ and $\beta_k = 2\beta_{k-1}$;
   – if $\beta_k \geqslant \bar{\beta} \rightarrow$ *Phase II*: *Step 2*;
   – otherwise – $(u^k, L(u^k), d^k) = (u^c, L(u^c), d^c)$
   – $\rightarrow$ *Phase II*: *Step 2.*

References

[1] Shapiro JF. A survey of Lagrangean techniques for discrete optimization. Annals of Discrete Mathematics 1979;5:113–38.
[2] Shapiro JF. Mathematical programming: structures and algorithms. New York: Wiley, 1979.
[3] Fisher ML. The Lagrangean relaxation method for solving integer programming problems. Management Science 1981;27:1–18.
[4] Fisher ML. An application oriented guide to Lagrangean relaxation. Interfaces 1985;15:10–21.
[5] Held M, Wolfe P, Crowder HP. Validation of subgradient optimization. Mathematical Programming 1974;6:62–88.
[6] Kiwiel KC. Method of descent for nondifferentiable optimization. Lecture Notes in Mathematics, vol. 33. Berlin: Springer, 1985.
[7] Lemarechal C. Constructing bundle methods for convex optimization. In: Hirriart-Hurruty JB, editor. Fermat days 1985: mathematics for optimization. Amsterdam: North-Holland, 1986. p. 201–40.
[8] Lemarechal C. Lagrangean decomposition and nonsmooth optimization: bundle algorithms, prox iteration, augmented Lagrangean. In: Giannessi F, editor. Nonsmooth optimization, methods and applications. Philadelphia: Gordon & Breach, 1992. p. 201–16.
[9] Hiriart-Urruty JB, Lemarechal C. Convex analysis and minimization algorithms, vols. I and II. Berlin: Springer, 1993.
[10] Shor NZ. Generalized gradients methods of nondifferentiable optimization employing space dilatation operations. In: Bachem A, Grotschel M, Korte B, editors. Mathematical programming. The state of the art. Bonn, 1982.
[11] Fumero F. Solving techniques for the Lagrangean dual. Quaderni DEP n.13, Dipartimento di Economia e Produzione, Politecnico di Milano, 1997.
[12] Poljak BT. Minimization of unsmooth functionals. U.S.S.R. Computational Mathematics and Mathematical Physics 1969;9:14–29.
[13] Goffin JL. On convergence rate of subgradient optimization methods. Mathematical Programming 1977;13:329–47.
[14] Allen E, Helgason R, Kennington J, Shetty B. A generalization of Polyak's convergence result for subgradient optimization. Mathematical Programming 1987;37:309–17.
[15] Kim S, Um B. Polyak's subgradient method with simplified projection for non-differentiable optimization with linear constraints. Optimization 1989;20:451–6.
[16] Kim S, Ahn H, Cho SC. Variable target value subgradient method. Mathematical Programming 1991;49:359–69.
[17] Fetterolf PC, Anandalingam G. A Lagrangean relaxation technique for optimizing interconnection of Local Area Networks. Operations Research 1992;40:678–88.
[18] Bazaraa MS, Goode JJ. A survey of various tactics for generating Lagrangian multipliers in the context of Lagrangian duality. European Journal of Operational Research 1979;3:322–38.
[19] Larsson T, Patriksson M, Stromberg AB. Conditional subgradient optimization – theory and applications. European Journal of Operational Research 1996;88:382–403.
[20] Camerini PM, Fratta L, Maffioli F. On improving relaxation methods by modified gradient techniques. Mathematical Programming Study 1975;3:26–34.
[21] Norkin VN. Method of nondifferentiable function minimization with averaging of generalized gradients. Kibernetika 1980;6:86–9.
[22] Kim S, Koh S, Ahn H. Two-direction subgradient method for non-differentiable optimization problems. Operations Research Letters 1987;6:43–6.
[23] Kim S, Koh S. On Polyak's improved subgradient method. Journal of Optimization Theory and Applications 1988;57(2):355–60.
[24] Kim S, Ahn H. Convergence of a generalized subgradient method for non-differentiable convex optimization. Mathematical Programming 1991;50:75–80.
[25] Sarin S, Karwan MH. Computational evaluation of two subgradient search methods. Computers and Operations Research 1987;14:241–7.
[26] Sarin S, Karwan MH, Rardin RL. A new surrogate-dual multiplier search procedure. Naval Research Logistics 1987;34:431–50.

[27] Kim S, Ahn H. Convergence properties of the modified subgradient method of Camerini et al. Naval Research Logistics 1990;37:961–6.
[28] Bazaraa MS, Sherali HD. On the choice of step size in subgradient optimization. European Journal of Operational Research 1981;7:380–8.
[29] Held M, Karp RM. The travelling salesman problem and minimum spanning trees. Operations Research 1970;18:1138–62.
[30] Barachet LL. Graphic solution of the travelling salesman problem. Operations Research 1957;5:841–5.
[31] Karg LL, Thompson GL. A heuristic approach to solving travelling salesman problems. Management Science 1964;10:225–48.
[32] Dantzig GB, Fulkerson DR, Johnson SM. Solution of a large scale travelling salesman problem. Operations Research 1954;2:393–410.
[33] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990;41:1069–72.
[34] Fumero F, Vercellis C. Capacity analysis in repetitive assemble-to-order manufacturing systems. European Journal of Operational Research 1994;78:204–15.
[35] Fumero F, Vercellis C. Capacity management through Lagrangean relaxation: an application to tires production. Production Planning and Control 1996;7:604–14.
[36] Fumero F, Vercellis C. Integrating distribution, machine assignment and lot-sizing via Lagrangean relaxation. International Journal of Production Economics 1997;49:45–54.

Francesca Fumero is Assistant Professor in Operations Research at Politecnico di Milano, Italy. She received her *Laurea* in Management Engineering from Politecnico di Milano and her Ph.D. in Computational Mathematics and Operations Research from Universitá degli Studi di Milano. Her main research and teaching interests concern models and algorithms for production and transportation problems, optimization techniques and decomposition algorithms, benchmarking and performance analysis via applications of Operations Research techniques. She is author of a number of papers published in international journals of the Operations Research field, such as *Transportation Science*, *European Journal of Operational Research*, *The International Journal of Production Economics*, *Production Planning and Control.*