Theory and Methodology

Lagrangean/surrogate relaxation for generalized assignment problems

Marcelo G. Narciso, Luiz Antonio N. Lorena *

INPE – Instituto Nacional de Pesquisas Espaciais, LAC, Caixa Postal 515, 12201 São José dos Campos SP, Brazil

Received 10 March 1997; accepted 7 January 1998

Abstract

This work presents Lagrangean/surrogate relaxation to the problem of maximum profit assignment of \( n \) tasks to \( m \) agents (\( n > m \)), such that each task is assigned to only one agent subject to capacity constraints on the agents. The Lagrangean/surrogate relaxation combines usual Lagrangean and surrogate relaxations relaxing first a set of constraints in the surrogate way. Then, the Lagrangean relaxation of the surrogate constraint is obtained and approximately optimized (one-dimensional dual). The Lagrangean/surrogate is compared with the usual Lagrangean relaxation on a computational study using a large set of instances. The dual bounds are the same for both relaxations, but the Lagrangean/surrogate can give improved local bounds at the application of a subgradient method, resulting in less computational times. Three relaxations are derived for the problem. The first relaxation considers a vector of multipliers for the capacity constraints, the second for the assignment constraints and the other for the Lagrangean decomposition constraints. Relaxation multipliers are used with efficient constructive heuristics to find good feasible solutions. The application of a Lagrangean/surrogate approach seems very promising for large scale problems. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Relaxation; Lagrangean and surrogate relaxation; Generalized assignment problem

1. Introduction

We examine in this paper Lagrangean/surrogate relaxation to the problem of maximum profit assignment of \( n \) tasks to \( m \) agents (\( n > m \)), such that each task is assigned to only one agent subject to capacity constraints on the agents. The Lagrangean/surrogate relaxation combines usual Lagrangean and surrogate relaxations relaxing first a set of constraints in the surrogate way. Then, the Lagrangean relaxation of the surrogate constraint is obtained and approximately optimized (one-dimensional dual). The problem considered is an important combinatorial optimization problem, a Generalized Assignment Problem (GAP).
Many real life applications can be modeled as a GAP, e.g., the resource scheduling, allocation of memory spaces, design of communication network with capacity constraints for each network node [2], assigning software development tasks to programmers, assigning jobs to computers in a network [2,9], vehicle routing problems [16], and others [6].

Cattrysse and Van Wassenhove [6] give a recent survey of algorithms for the problem. Amini and Racer [1] present a computational comparison of alternative solution methods. Although GAP is NP-hard [19], the preponderance of algorithms in literature are exact tree-search methods [6], and there are few known efficient heuristics for the problem [1,6,7,17,23,28,32,35,36].

This work complements the results presented in the previous paper of Lorena and Narciso [35] comparing the use of Lagrangean/surrogate relaxation and heuristics. Three relaxations are derived for the problem. The first relaxation considers a vector of multipliers for the capacity constraints, the second for the assignment constraints and the other for the Lagrangean decomposition constraints. Relaxation multipliers are used with efficient constructive heuristics to find good feasible solutions. The heuristics are tested and compared using instances of problems from the literature.

Section 2 gives a formal description of the new Lagrangean/surrogate relaxation to 0-1 linear programming problem. It is proposed to solve the Lagrangean/surrogate dual using the subgradient method, widely known in the Lagrangean relaxation context. Although it cannot improve the dual bounds when compared to the usual Lagrangean dual, it can produce improved local bounds, calculated by a naive search procedure that approximately solves the one-dimensional dual problem generated by the Lagrangean/surrogate relaxation. The three Lagrangean/surrogate relaxations for the GAP are presented in Section 3. Section 4 describes a general relaxation heuristic, that uses alternately each Lagrangean/surrogate relaxation and the traditional Lagrangean and Lagrangean decomposition relaxations. The relaxation solutions are made feasible to the GAP using efficient constructive heuristics proposed for each relaxation. Section 5 presents several computational tests with problems of the literature. In Section 6 we conclude with some comments.

2. Lagrangean/surrogate relaxation

The Lagrangean relaxation has a successful outline of applications in the Operational Research literature. Beginning with the classical paper of Held and Karp [26] for the Traveling Salesman problem, the number of applications using Lagrangean relaxations continually increases, and the task of collecting a significant group of published applications is now a hard one. Survey papers [15,20] and books [38,42] give a formal description of Lagrangean relaxation.

Depending on the set of constraints relaxed, the bounds obtained can be improved. Theoretical results are found in several papers [15,20,24]. The bounds are frequently incorporated in branch and bound like procedures [40].

The Lagrangean decomposition is a variable splitting technique that extends the use of the traditional Lagrangean relaxation and (in general) improves the relaxation bounds [24,28] at a corresponding increase in computational times.

The Lagrangean bounds are improved solving the conformably Lagrangean dual. A subgradient method makes a controlled search, providing a sequence of relaxation values with simple convergence properties [14,39].

The surrogate relaxation is not well known and applied as the Lagrangean one [13,21,22,29]. See [38] for a recent book describing the Lagrangean and surrogate relaxations.

The Lagrangean/surrogate relaxation is a new proposal that combines the two relaxations, Lagrangean and surrogate. It was considered to prevent the main drawback of the subgradient like methods, the very oscillating behavior at the first iterations in the sequence of Lagrangean bounds.
The Lagrangean/surrogate was used recently in other applications, such that for location problems [41]. It started in the applications for multidimensional knapsack problems [18], set covering problems [34], and our early paper on GAP [35]. For the problems and instances studied, the performance of surrogate heuristics improved almost twice the computational times of the corresponding Lagrangean heuristics, with same quality for the bounds obtained.

In general terms, suppose the following 0-1 linear programming problem:

\[
v(P) = \text{Max } cx \\
\text{subject to } Ax \leq b, \\
Dx \leq e, \\
x \in \{0, 1\}^n,
\]

where \(c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, D \in \mathbb{R}^{p \times n} \) and \(e \in \mathbb{R}^p\). At this point it is only necessary to think of \(Dx \leq e\) as the easily enforced constraints and \(Ax \leq b\) as the complicating ones. Defining \(T = \{x \in \{0, 1\}^n | Dx \leq e\}\), for a given multiplier \(\omega \in \mathbb{R}^m_+\) the Lagrangean relaxation of (P) is as follows.

\[
(LP^\omega) \quad v(LP^\omega) = \text{Max}\{cx - \omega (Ax - b)\}
\]

subject to \(x \in T\).

The Lagrangean dual is then the optimization problem in \(\omega\),

\[
(D^\omega) \quad v(D^\omega) = \text{Min}\{v(LP^\omega)\}
\]

subject to \(\omega \in \mathbb{R}^m_+\).

A very popular optimization method to solve (D\(^\omega\)) is the subgradient optimization, proposed by Poljak [39] in 1969, and first used in the Lagrangean relaxation context by Held and Karp [26]. A subgradient is directly identified after solving (LP\(^\omega\)) as \(g^\omega = Ax^\omega - b\), where \(x^\omega\) is an optimal solution of (LP\(^\omega\)). The subgradient method updates the multiplier \(\omega_k\) as \(\omega_{k+1} = \omega_k + \theta_k g^\omega_k\), and the following rules for the step sizes establish well-known convergence conditions [14,39]:

(i) \(\theta_k > 0, \lim_{k \to \infty} \theta_k ||g^\omega_k|| = 0 \) and \(\sum_{k=1}^{\infty} \theta_k ||g^\omega_k|| = \infty\) ([14]); or

(ii) \(\theta_k = \beta_k [v(LP^\omega) - v(D^\omega)] / ||g^\omega_k||^2\), where \(0 < \delta < \beta_k \leq 2 - \delta\) ([39]).

Held et al. [27] validated other step sizes suggested by Held and Karp [26], were in expression (ii) the value \(v(D^\omega)\) is substituted by a lower bound (a feasible solution to (P)), and parameter \(\beta_k\) varying at the interval \(0 \leq \beta_k \leq 2\), beginning with \(\beta_k = 2\). These step sizes although widely used, do not satisfy the second condition on expression (i).

Other subgradient methods appeared in literature [4,5,8,30,31]. More elaborated, they increase the local computational times computing descent directions [8], or combining subgradients of previous iterations [4,5], or realizing projections onto general convex sets [30,31,33]. Experimental results with some of these methods show an improvement in performance compared to the subgradient method [30,33]. The subgradient method remains the widely used approach in the Lagrangean relaxation context.

We introduce in the sequel the Lagrangean/surrogate relaxation of (P). First the constraints \(Ax \leq b\) are relaxed in a surrogate way [21]. For a given \(\lambda \in \mathbb{R}^m_+\), the surrogate problem of (P) is

\[
v(SP^\lambda) = \text{Max } cx \\
\text{subject to } \lambda(Ax - b) \leq 0, \\
x \in T.
\]

Frequently (SP\(^\lambda\)) is a difficult problem (like (P)). A linear programming version (surrogate continuous) (SP\(^\lambda\))\_linear of (SP\(^\lambda\)) can be obtained substituting \(T\) by \(T\_\text{linear} = \{x \in [0,1]^n | Dx \leq e\}\) in (SP\(^\lambda\)).

\[\text{SP}^\lambda \in \mathbb{R}^m_+ \]
Consider a relaxation in the Lagrangean way of problem \((SP^k)\). Given \(\lambda \in \mathbb{R}_m^m\), and a parameter \(t \geq 0\), the Lagrangean/surrogate relaxation of \((P)\) is

\[
(L_tSP^k) \quad v(L_tSP^k) = \max cx - t \cdot \lambda(Ax - b)
\]

subject to \(x \in T\).

A Lagrangean/surrogate dual can be identified here as

\[
(D_t^k) \quad v(D_t^k) = \min \{v(L_tSP^k)\}
\]

subject to \(t \cdot \lambda \in \mathbb{R}_m^m\).

It is immediate that setting \(\omega = t \cdot \lambda\), problem \((D_t^k)\) is the Lagrangean dual \((D^\omega)\), and the optimal lower bound limits coincide for both Lagrangean and Lagrangean/surrogate approach.

A local dual can be identified using the Lagrangean/surrogate relaxation. Suppose \(\lambda\) fixed, the one-dimensional dual in \(t\) is

\[
(D_t^k) \quad v(D_t^k) = \min \{v(L_tSP^k)\}
\]

subject to \(t \geq 0\).

When set \(T\) has the integrality property \(v(D_t^k) = v(SP^k)_{\text{linear}} [20]\). In general we have \(v(L_tSP^k) \geq v(D_t^k) \geq v(SP^k) \geq v(P)\). The attractive characteristic of relaxation \((L_tSP^k)\) is that for \(t = 1\) we have the usual Lagrangean relaxation using the multiplier \(\lambda\). An exact solution to \((D_t^k)\) may be obtained by a search over different values of \(t\) (see [25,37]). However we have a range of values \(t_0 \leq t^* \leq t_1\) with \(t_0 = 1\) or \(t_1 = 1\) which also produces improved bounds (for \(t_1 = 1\) see Fig. 1).

For a fixed \(\lambda\), the following inequalities are valid

\[
v(L_1SP^k) \geq v(L_{t^*}SP^k) \geq v(D_t^k) \geq v(SP^k) \geq v(P).
\]

Considering the application of a subgradient method, the same convergence rules (i) and (ii) can be directly identified when the multiplier \(t^* \lambda\) is used. Locally the Lagrangean/surrogate can provide better bounds than their Lagrangean (alone) counterpart. Using the Lagrangean/surrogate bound \(v(L_tSP^k)\) at the Held and Karp step size rule can be attractive. The subgradient direction \(g_{t^*}\) obtained from problem \((L_tSP^k)\) can give a different direction of the subgradient obtained when problem \((L_1SP^k)\) is solved. Therefore different se-

![Fig. 1. Lagrangean/surrogate bounds.](image-url)
quences of relaxation bounds can be obtained beginning with the same initial multiplier \( \lambda \). The computational tests of Section 5 compare the relaxations, using a subgradient method that considers the updating rules:

\[
\lambda_{k+1} = \lambda_k + \beta_k [v(L_k SP) - v_f] g^2_k / \|g^2_k\|^2, \quad 0 \leq \beta_k \leq 2
\]

\( v_f \) the value of a feasible solution to \((P)\).

In fact, we are updating indirectly the multipliers \( t^i \lambda \). Suppose \( t^i \) calculated at the first iteration of the subgradient algorithm and fixed at remaining iterations, a correspondingly updating rule for multipliers \( t^i \) is obtained substituting \( \beta_k \) by \( t^i \cdot \beta_k \) \( (t^i \) is only a scale factor). At the other extreme, an approximate effect can be seen when \( t^i \) is calculated at each subgradient iteration. The set \( T \) has the integrality property at the Lagrangean relaxations used in the applications on set covering [34] and GAP [10–12,35], and \( t^i \) was calculated at each iteration of the subgradient method solving \((SP)_{linear}\). For the two problems studied, the linear relaxation of \((SP)\) induced a stability at the sequence of relaxations that resulted in an improvement in computational times, compared with the application of the ordinary Lagrangean relaxation for the same set of instances.

In this paper we propose to use three Lagrangean relaxations for GAP, only one having the integrality property (the same used in [35]). The corresponding Lagrangean/surrogate are derived, and an intermediate approach is used to calculate \( t^i \). A naive line search is used to calculate a \( t^i \) belonging to the interval \( t_0 \leq t^i \leq t_1 \) \( (t_0 = 1 \) or \( t_1 = 1 \). The local optimization procedure \( \text{search}_{t^i} \) described in the following contributes for the reduction in the oscillating behavior of the sequence \( \{v(L_{t^i} SP(i))\} \), but it increases the local computational times resulting in \( k^i \) evaluations of \( v(L_{t} SP) \). The question here is how to transform that Lagrangean/surrogate characteristic on a gain at the overall computational times. The immediate answer is make \( \text{search}_{t^i} \) only when necessary. The oscillating behavior of subgradient sequences is very accentuated at the first steps, then we propose to use \( \text{search}_{t^i} \) to the point that the same \( t^i \) is found for a number \( (n\_consec) \) of consecutive \( \lambda \). The computational tests of Section 5 confirm the feasibility of this approach for a large set of GAP instances.

For a fixed \( \lambda \), \( \text{search}_{t^i} \) is described by the following pseudo-code:

\[
\text{Search}_{t^i}
\]

Given \( s \) \( (\text{the initial step size}) \),

\( k_{\text{max}} \) \( (\text{the maximum number of iterations}) \),

Set \( t := 0 \) \( (\text{the current Lagrangean/surrogate multiplier}) \),

\( v^* := v(L_0 SP) \) \( (\text{the current best upper bound}) \),

\( k^i := 0 \) \( (\text{the current number of iterations}) \);

Repeat

Set \( t := t + s \),

\( k^i := k^i + 1 \);

If \( k^i > k_{\text{max}} \) then stop

Else solve \((L^i SP)\)

End_If

If \( v(L_{t} SP) < v^* \) then

Set \( t^i = t \),

\( v^* := v(L_{t} SP) \);

Calculate \( \mu^i = \lambda \cdot (Ax^i - b) \) \( (\mu^i \) is the slope of the Lagrangean/surrogate dual function);

If \( \mu^i > 0 \) then solve \((L_{t-\mu^i/2} SP)\) \( \text{(try to improve the current multiplier returning half step size)} \)

If \( v(L_{t-s/2} SP) < v(L_{t} SP) \) then \( t^i := t - s/2 \)

End_If
Stop;  
End_If
Else
Solve \((L_{t-\lambda/2}SP^\lambda)\),  
If \(v(L_{t-\lambda/2}SP^\lambda) < v(L_\lambda SP^\lambda)\) then \(t^* := t - s/2\)
End_If
If \(k^* \leq \lceil k_{\max}/2 \rceil\) then set \(s := s/2\) (the initial step size is too large and must be halved for the next procedure applications);  
End_If  
Stop;  
End_If
Until (stop conditions).

This search procedure results in a multiplier \(t^*\) that is then used in the Lagrangean/surrogate relaxation given the value \(v(L_\lambda SP^\lambda)\).

3. The new relaxations used for the GAP

The GAP is best described using knapsack problems [36]. Given \(n\) items and \(m\) knapsacks, with \(p_{ij}\) as the profit of assigning item \(j\) to knapsack \(i\), \(w_{ij}\) as the weight of assigning item \(j\) to knapsack \(i\), and \(c_i\) the capacity of knapsack \(i\), assign each item \(j\) to exactly one knapsack \(i\), not exceeding knapsack capacities. Then GAP can be formulated as follows.

\[
v(GAP) = \text{Max} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \\
(\text{GAP}) \quad \text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_{ij} \leq c_i, \quad i \in M = \{1, \ldots, m\}, \\
\sum_{i=1}^{m} x_{ij} = 1, \quad j \in N = \{1, \ldots, n\}, \\
x_{ij} \in \{0, 1\}, \quad i \in M, \ j \in N.
\]

For given \(t\) and \(\lambda\), the following Lagrangean/surrogate relaxations of (GAP) can be defined:

(a) Relaxing the capacity constraints (for \(t \geq 0\) and \(\lambda \in \mathbb{R}^m_+\) see [6]):

\[
v(L_tSP^\lambda)_a = \text{Max} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} - t \cdot \lambda_i w_{ij}) x_{ij} + t \cdot \sum_{i=1}^{m} \lambda_i c_i \\
(L_tSP^\lambda)_a \quad \text{subject to} \quad \sum_{i=1}^{m} x_{ij} = 1, \quad j \in N, \\
x_{ij} \in \{0, 1\}, \quad i \in M, \ j \in N;
\]

(b) Relaxing the assignment constraints (for \(t \in \mathbb{R}\) and \(\lambda \in \mathbb{R}^n\) see [6]):

\[
v(L_tSP^\lambda)_b = \text{Max} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} + t \cdot \lambda_j) x_{ij} - t \cdot \sum_{j=1}^{n} \lambda_j \\
(L_tSP^\lambda)_b \quad \text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_{ij} \leq c_i, \ i \in M, \\
x_{ij} \in \{0, 1\}, \quad i \in M, \ j \in N; \ and
(c) Relaxing the variable splitting constraints (for \( t \in \mathbb{R}, \gamma + \delta = 1 \) and \( \lambda \in \mathbb{R}^{m \times n} \) see [28]):

\[
\begin{align*}
\text{v} & (L_t\text{SGAP}^i)_{c} = \max \quad \gamma \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}x_{ij} + \delta \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}y_{ij} + t \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_{ij}(x_{ij} - y_{ij}) \\
& \text{subject to} \quad \sum_{j=1}^{m} w_{ij}x_{ij} \leq c_{i}, \quad i \in M, \\
& \quad \sum_{i=1}^{m} y_{ij} = 1, \quad j \in N, \\
& \quad x_{ij}, y_{ij} \in \{0, 1\}, \quad i \in M, \quad j \in N.
\end{align*}
\]

For fixed \( t \) and \( \lambda \), \( (L_t\text{SGAP}^i)_{a} \) is easily solved by setting \( x'_{ij} = 1 \) for \( i' = \text{index of max} \{ p_{ij} - t \cdot \lambda_{ij} \}, \quad j \in N \), and \( x_{ij} = 0 \) otherwise. \( (L_t\text{SGAP}^i)_{b} \) is not so easily solved as it separates in \( m \)-0-1 knapsack problems that must be optimally solved. The algorithm of Horowitz and Sahni (see [36]) was used in the computational tests of Section 5. \( (L_t\text{SGAP}^i)_{c} \) is solved observing that it can be decomposed in the following problems:

\[
\begin{align*}
\text{v} & (L_t\text{SGAP}^i)_{c}^{X} = \max \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (\gamma \cdot p_{ij} + t \cdot \lambda_{ij})x_{ij} \\
& \text{subject to} \quad \sum_{j=1}^{m} w_{ij}x_{ij} \leq c_{i}, \quad i \in M, \\
& \quad x_{ij} \in \{0, 1\}, \quad i \in M, \quad j \in N; \\
\text{v} & (L_t\text{SGAP}^i)_{c}^{Y} = \max \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (\delta \cdot p_{ij} - t \cdot \lambda_{ij})y_{ij} \\
& \text{subject to} \quad \sum_{i=1}^{m} y_{ij} = 1, \quad j \in N, \\
& \quad y_{ij} \in \{0, 1\}, \quad i \in M, \quad j \in N.
\end{align*}
\]

We have \( \text{v} (L_t\text{SGAP}^i)_{a} = \text{v} (L_t\text{SGAP}^i)_{c}^{X} + \text{v} (L_t\text{SGAP}^i)_{c}^{Y} \). Problems \( \text{v} (L_t\text{SGAP}^i)_{c}^{X} \) and \( \text{v} (L_t\text{SGAP}^i)_{c}^{Y} \) are similar to relaxations (b) and (a), respectively and are solved using the procedures above.

4. The General Relaxation Heuristic for GAP

The relaxations presented in Section 3 are integrated with an outer optimization problem that looks for the best relaxation values varying the \( \lambda \) values. In the context of duality we are solving the dual problem

\[
\begin{align*}
(D) \quad & v(D) = \min \text{v} (\text{REL}^\lambda_{t}) \\
& \text{subject to} \quad \lambda \in A,
\end{align*}
\]

where, for a fixed \( t \), \( (\text{REL}^\lambda_{t}) \) can be one of the three relaxations \( (L_t\text{SGAP}^i)_{a}, (L_t\text{SGAP}^i)_{b}, \) or \( (L_t\text{SGAP}^i)_{c} \) and \( A \) the sets \( \mathbb{R}^{m}, \mathbb{R}^{n} \) or \( \mathbb{R}^{m \times n} \), respectively.

To solve problem (D), a subgradient algorithm has been used with success in applications to set covering problems [34], 0-1 multiknapsack problems [18], location problems [41], and in our early application to GAP [35]. In the applications on set covering, 0-1 multiknapsack problems and GAP, the relaxation used is a linear relaxation of the corresponding surrogate relaxation. Each relaxation solution was made feasible using special proposed constructive heuristics.

The Relaxation Heuristic (RH) presented below uses a general relaxation \( (\text{REL}^\lambda_{t}) \) that can be one of the relaxations described in the Section 3.
Relaxation Heuristic (RH)

Given \( \lambda \);
Set \( lb := -\infty \), \( ub := +\infty \);
Repeat
Solve relaxation (REL)\(^{\lambda} \) obtaining \( x^{\lambda} \) and \( v(\text{REL}^{\lambda}) \);
Obtain a feasible solution \( x_f \) and \( v_f = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_{f_{ij}} \),
applying a Constructive Heuristic using \( \lambda \) and/or \( x^{\lambda} \);
\( lb := \min \{ lb, v_f \} \);
\( ub := \min \{ ub, v(R_t^{\lambda}) \} \);
Update the subgradient direction \( g^{\lambda} \), the step size \( \theta \) and the multiplier \( \lambda \);
Make stopping tests;
Until (stop conditions).

Some steps in (RH) must be explained. The heuristic will be specialized defining the relaxation (REL\(^{\lambda} \)).
There are six possibilities for the Lagrangean/surrogate relaxations defined in section three, alternating the value of parameter \( t \) that can be fixed to 1 (Lagrangean case) or equal to \( t^\ast \) obtained in search\(_t^\ast \) (Lagrangean/surrogate case). There are some common and some specific steps.
The common steps are:
The step size used is \( \theta = \beta(ub - lb)/||g^{\lambda}||^2 \). The control parameter \( \beta \) is the Held and Karp [26] control that makes \( 0 \leq \beta \leq 2 \), beginning with \( \beta = 2 \). If after 20 iterations \( ub \) not decreases, \( \beta \) is updated to \( \beta = \beta/2 \);
The stopping tests used are:
(i) number of iterations greater than 600;
(ii) \( \beta \leq 0.005 \);
(iii) \( ub - lb < 1 \); and
(iv) \( ub \) (integer part of the upper bound) unchanged by 20 consecutive part.
The specific steps are:
For (REL\(^{\lambda} \)) = (L\(_1\)SGAP\(^{\lambda} \)) or (L\(_2\)SGAP\(^{\lambda} \)):
(i) The initial \( \lambda \) used is \( \lambda_i = (\sum_{j=1}^{n} w_{ij} - c_i) / (\sum_{j=1}^{n} w_{ij}), \ i \in M \);
(ii) the subgradient direction is \( g^{\lambda}_i = c_i - \sum_{j=1}^{n} w_{ij} x^{\lambda}_{ij} \), \( i \in M \);
(iii) the updating of \( \lambda \) is \( \lambda_i = \max \{ 0, \lambda_i + \theta g^{\lambda}_i \}, \ i \in M \);
(iv) the Constructive Heuristic used is basically the heuristic (CH) of Lorena and Narciso [35]. (CH) was divided in three parts (CH1), (CH2), and (CH3). In (CH1) the relaxation solution is modified trying to make feasible the capacity constraints maintaining the feasibility of the assignment constraints.
If (CH1) fails in finding a feasible solution, in (CH2) the relaxation solution is used again, making feasible the capacity constraints while trying to maintain the feasibility of the assignment constraints.
If the two phases fail (end without a feasible solution), (CH3) tries to construct directly a feasible solution using \( \lambda \). The following additional part is included and located in sequence after (CH1).

\{(CH1)\}a-Try to make feasible the capacity constraints maintaining the feasibility of the assignment constraints
Set \( z_{ij} := x_{ij}^\lambda; \ i = 1, \ldots, m \) (\( z \) receives the relaxation solution \( x^\lambda \))
\( knap_i := c_i - \sum_{j=1}^{n} w_{ij} z_{ij}, \ i = 1, \ldots, m; \)
For \( i = 1, 2, \ldots, m \) do
If \( knap_i < 0 \) then
\( j^\ast := \text{index of Min} \{ p_{ij}/w_{ij} | z_{ij} = 1 \} \)
\( K := \{ k | (knap_k - w_{ij}) \geq 0; \ q = 1, 2, \ldots, m; \ q \neq i \} \)
If \( K \neq \emptyset \) then
instances, with sizes \( \{5, 8, 10\} \) and that considers now the ratios \( p_j/w_{ij} \). (CH1a) improves in almost 20% the bounds quality in computational tests. At the end of each part the feasible solution obtained may be improved by a sequence of local searches proposed by Martello and Toth [36] (see also [35]).

For \( (REL) \) = \( (L_1SGAP^a) \), or \( (L_1SGAP^b) \),

(i) The initial \( \lambda \) used is \( \lambda_j = 1, j \in N \);  
(ii) the subgradient direction is \( g_j = \sum_i \lambda_{ij} x_{ij} - 1, j \in N \);  
(iii) the updating of \( \lambda \) is \( \lambda_j = \lambda_j + \theta g_j, j \in N \);  
(iv) the Constructive Heuristic used here can be one of the two used before. Solving \( (L_1SGAP^c) \), we solve problems similar to \( (L_1SGAP^a) \) and \( (L_1SGAP^b) \). Using as a base for comparison the computational tests conducted we decided to use the same heuristic used for \( (L_1SGAP^a) \), with part 3 modified accordingly to conform the dimension of the multiplier \( \lambda \in \mathbb{R}^{m \times n} \).

The procedure search_\( t^* \) of Section 3 is used in \( (L_1SGAP^a) \), \( (L_1SGAP^b) \) and \( (L_1SGAP^c) \) with the following parameters: \( s = 0.5, k_{\text{max}} = 5 \) and \( n_{\text{consec}} = 10 \). They arise to be better than other parameter values adopted. As we are fixing the value of \( t^* \) after \( n_{\text{consec}} \) iterations, there is no motive to use an exact algorithm in substitution to search_\( t^* \), as it results in lost effort when the subgradient sequences converge.

5. Computational results

The heuristics were first tested on a set of 12 small instances used before in [35], with sizes \( m \times n \), for \( m \in \{5, 8, 10\} \) and \( n \in \{15, 20, 24, 25, 30, 32, 40, 48, 50, 60\} \). Then the tests continued with a set of 24 large scale instances, with sizes \( m \in \{5, 10, 20\} \) and \( n \in \{100, 200\} \). The data were divided in four classes \( A, B, C \) and
D, and were obtained from the OR-library [3] (only class C for the small instances and classes A, B, C and D for the large instances). Problems of classes A, B and C present increasing knapsacks. Class D is in theory composed with (the most) difficult correlated problems.

Table 1 presents computational results using the 12 small instances. Only class C was used (the unique class with known optimal solutions), and the results are mean values for 5 replications in each problem. Number in lines represent mean values. The computational times are obtained using a PC486DX2, and all the algorithms are coded in C language. Notation in columns follow.

- **times**: final times in seconds of proposed heuristics;
- **iter.**: number of Lagrangean relaxations solved;
- **gap1**: \((\text{optimal solution} - \text{lb}) / \text{optimal solution}\);
- **gap2**: \((\text{ub} - \text{optimal solution}) / \text{optimal solution}\);
- **gap2 = 5%,...,0.5%**: computational times to gap2 attain 5%, 4%, 3%, 2%, 1% and 0.5%.

When the percentages are reached, they reflect the performance of the relaxation without the influence of the stopping tests. The numbers in parenthesis represent the number of problems the corresponding relaxations have reached the admitted percentage for gap2.

For the 24 large scale instances, the results are reported in Table 2. Number in lines represent mean values. The computational times are in SUN SPARC 5 s.

The results in tables are obtained with the 600 iterations limit. To compute the gaps 1 and 2 we need the optimal solution that was not previously known for the large scale instances. The values used was then obtained running \((RH)\) for each Lagrangean/surrogate relaxation and instance without limit of iterations, searching the optimality condition \((\text{ub} - \text{lb}) < 1\). The best feasible solution obtained are (solutions signed by (*) are optimal):

**Table 1**

<table>
<thead>
<tr>
<th>Relaxation</th>
<th>Times (s)</th>
<th>Iter.</th>
<th>gap1 ((\times 10^{-3}))</th>
<th>gap2 ((\times 10^{-3}))</th>
<th>gap2 ((= 5%))</th>
<th>gap2 ((= 4%))</th>
<th>gap2 ((= 3%))</th>
<th>gap2 ((= 2%))</th>
<th>gap2 ((= 1%))</th>
<th>gap2 ((= 0.5%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((L_1SGAP)_a)</td>
<td>1.99</td>
<td>300</td>
<td>3.87</td>
<td>11.16</td>
<td>0.10 (12)</td>
<td>0.11 (12)</td>
<td>0.14 (12)</td>
<td>0.16 (10)</td>
<td>0.50 (7)</td>
<td>0.89 (1)</td>
</tr>
<tr>
<td>((L_1SGAP)_b)</td>
<td>1.12</td>
<td>196</td>
<td>5.28</td>
<td>11.24</td>
<td>0.09 (12)</td>
<td>0.09 (12)</td>
<td>0.10 (12)</td>
<td>0.10 (10)</td>
<td>0.16 (7)</td>
<td>0.26 (1)</td>
</tr>
<tr>
<td>((L_2SGAP)_a)</td>
<td>12.32</td>
<td>192</td>
<td>0.84</td>
<td>1.76</td>
<td>2.59 (12)</td>
<td>2.81 (12)</td>
<td>3.09 (12)</td>
<td>3.48 (12)</td>
<td>4.30 (12)</td>
<td>6.09 (12)</td>
</tr>
<tr>
<td>((L_2SGAP)_b)</td>
<td>15.18</td>
<td>248</td>
<td>0.90</td>
<td>1.82</td>
<td>2.24 (12)</td>
<td>2.38 (12)</td>
<td>2.61 (12)</td>
<td>3.31 (12)</td>
<td>7.52 (12)</td>
<td>8.65 (12)</td>
</tr>
<tr>
<td>((L_3SGAP)_a)</td>
<td>46.45</td>
<td>444</td>
<td>0.95</td>
<td>4.76</td>
<td>4.12 (12)</td>
<td>6.18 (12)</td>
<td>9.72 (12)</td>
<td>16.76 (12)</td>
<td>21.32 (11)</td>
<td>24.29 (8)</td>
</tr>
<tr>
<td>((L_3SGAP)_b)</td>
<td>28.62</td>
<td>370</td>
<td>0.37</td>
<td>2.38</td>
<td>2.11 (12)</td>
<td>2.98 (12)</td>
<td>4.76 (12)</td>
<td>9.60 (12)</td>
<td>10.58 (12)</td>
<td>10.60 (12)</td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>Relaxation</th>
<th>Times (s)</th>
<th>Iter.</th>
<th>gap1 ((\times 10^{-3}))</th>
<th>gap2 ((\times 10^{-3}))</th>
<th>gap2 ((= 5%))</th>
<th>gap2 ((= 4%))</th>
<th>gap2 ((= 3%))</th>
<th>gap2 ((= 2%))</th>
<th>gap2 ((= 1%))</th>
<th>gap2 ((= 0.5%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((L_1SGAP)_a)</td>
<td>9.15 (22)</td>
<td>232</td>
<td>1.58</td>
<td>2.23 (22)</td>
<td>1.50 (22)</td>
<td>1.55 (22)</td>
<td>1.60 (22)</td>
<td>1.66 (22)</td>
<td>1.83 (21)</td>
<td>2.04 (19)</td>
</tr>
<tr>
<td>((L_1SGAP)_b)</td>
<td>2.60 (24)</td>
<td>148</td>
<td>2.48</td>
<td>2.45 (24)</td>
<td>0.80 (24)</td>
<td>0.80 (24)</td>
<td>0.80 (24)</td>
<td>0.80 (24)</td>
<td>0.86 (23)</td>
<td>0.88 (21)</td>
</tr>
<tr>
<td>((L_2SGAP)_a)</td>
<td>184.54 (17)</td>
<td>534</td>
<td>0.44</td>
<td>2.24 (17)</td>
<td>35.88 (17)</td>
<td>37.6 (17)</td>
<td>39.63 (17)</td>
<td>44.64 (16)</td>
<td>48.55 (16)</td>
<td>52.37 (16)</td>
</tr>
<tr>
<td>((L_2SGAP)_b)</td>
<td>115.83 (23)</td>
<td>433</td>
<td>0.30</td>
<td>1.00 (23)</td>
<td>14.15 (23)</td>
<td>14.76 (23)</td>
<td>15.51 (23)</td>
<td>17.43 (23)</td>
<td>19.20 (23)</td>
<td>22.53 (22)</td>
</tr>
<tr>
<td>((L_3SGAP)_a)</td>
<td>248.44 (13)</td>
<td>600</td>
<td>2.50</td>
<td>16.12 (13)</td>
<td>35.95 (13)</td>
<td>42.72 (13)</td>
<td>24.57 (9)</td>
<td>30.60 (8)</td>
<td>23.59 (5)</td>
<td>19.52 (4)</td>
</tr>
<tr>
<td>((L_3SGAP)_b)</td>
<td>278.83 (21)</td>
<td>541</td>
<td>1.18</td>
<td>4.94 (21)</td>
<td>9.80 (21)</td>
<td>11.41 (20)</td>
<td>8.06 (19)</td>
<td>9.90 (18)</td>
<td>8.39 (15)</td>
<td>9.61 (13)</td>
</tr>
</tbody>
</table>
Some comments and conclusions can be derived from Tables 1 and 2:

(a) *The application on small instances (Table 1):* Initially we compare the results with the ones in [35]. Lorena and Narciso [35] compared their heuristics with the ones of Martello and Toth [36] and of Klastorin [32]. For the class C the best gap \( \hat{g}_1 \approx 0.4\% \) was obtained using the surrogate heuristic. The results on Table 1 for the corresponding relaxations \( L_1SGAP^k_a \) and \( LtSGAP^k_a \) conserve this result at approximately 0.38\% and 0.52\%, respectively. There is a large improvement to 0.03\% using the relaxation \( LtSGAP^k_c \). The other relaxations also improve the gap1 to 0.08\%, 0.09\% and 0.09\%, respectively.

Comparing the relaxations bounds \( L_1SGAP^k_a \) and \( LtSGAP^k_a \) are the faster ones but also the worse ones in terms of gaps. \( L_1SGAP^k_b \) and \( LtSGAP^k_b \) are faster than \( L_1SGAP^k_c \) and \( LtSGAP^k_c \). They are also almost equivalent in terms of gaps.

Comparing the Lagrangean and Lagrangean/surrogate. As the bounds (gap1 and gap2) obtained are almost the same for both versions, the last six columns will give the difference in terms of time to reach a pre-defined gap2 percentage (5\%, 4\%, 3\%, 2\%, 1\% and 0.5\%). The mean (of the six columns) for the six relaxations (by line) are 0.32, 0.13, 2.73, 4.62, 13.81 and 6.77, respectively. Then, the Lagrangean/surrogate reach almost the same results as the Lagrangean heuristic in 41\%, 123\% and 69\% of time. The Lagrangean/surrogate found 5 percentages more (sum of the number in parenthesis) than the Lagrangean one: 5 for \( LtSGAP^k_c \).

(b) *The application on large instances (Table 2):* The gaps obtained for the large scale instances are almost equivalent for all relaxations, and of very good quality. We can point out the gap1 = 0.03\% and the gap2 = 0.1\% for \( LtSGAP^k_c \).

The mean of the six last columns for the six relaxations (by line) are 1.7, 0.82, 43.11, 17.26, 29.54 and 9.52, respectively. Then, the Lagrangean/surrogate reach almost the same results than the Lagrangean heuristic in 48\%, 40\% and 32\% of time. The Lagrangean/surrogate found more 105 percentages (sum of the number in parenthesis) than the Lagrangean one: 12 for \( LtSGAP^k_d \), 38 for \( LtSGAP^k_b \) and 55 for \( LtSGAP^k_c \).

For the instances tested, the Lagrangean/surrogate heuristic is more stable and can reach very good bounds with about 32\% of the corresponding time required by the Lagrangean heuristic. It appears to be indicated for large scale instances when high quality results are searched.

6. Conclusion

We proposed Lagrangean/surrogate heuristics for generalized assignment problems. The computational results show that the Lagrangean/surrogate can be better than the corresponding Lagrangean relaxation using the subgradient optimization method, mainly for application with large scale instances. This fact was already noted in other combinatorial optimization problems [18,34,35,41]. The local optimization induced by the Lagrangean relaxation of the surrogate constraints induce fast convergence of the sequence of
relaxation values in (RH). The local search depends on a number of parameters that can influence the performance of the Lagrangean/surrogate heuristic. The local optimization also produces a (different) subgradient that can be used with other subgradient methods [4,5,30,33], a possible extension of the current approach.

A large number of instances were employed in computational tests with three Lagrangean/surrogate relaxations, including the variable splitting (decomposition). For the large scale problems we reported 15 optimal solutions obtained running the Lagrangean/surrogate heuristics without iterations limit. For the results reported in tables the iteration limit was fixed to 600 consecutive Lagrangeans solved, as we are comparing computational times to reach pre-defined small gaps. The Lagrangean/surrogate heuristic was more stable and can reach very good bounds with about 32% of the corresponding time required by the usual Lagrangean heuristic. It appears to be indicated for large scale instances when high quality results are searched.

This work contributes proposing a new relaxation heuristic with very good performance in large scale instances of GAPs.

Acknowledgements

The second author acknowledges Conselho Nacional de Desenvolvimento Científico e Tecnológico CNPq (proc. 520844/96-3) for partial financial support. We greatly acknowledge two anonymous referees for helpful suggestions and comments.

References
