A hybrid search combining interior point methods and metaheuristics for 0-1 programming

Agnès Plateau, Dominique Tachat and Pierre Tolla

LAMSADÉ, UPRES-A 7024, Université Paris Dauphine, place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16 France
E-mail: plateau@lamsade.dauphine.fr

Received 4 October 2000; received in revised form 15 April 2002; accepted 20 April 2002

Abstract

Our search deals with methods hybridizing interior point processes and metaheuristics for solving 0-1 linear programs. This paper shows how metaheuristics can take advantage of a sequence of interior points generated by an interior point method. After introducing our work field, we present our hybrid search which generates a diversified population. Next, we explain the whole method combining the solutions encountered in the previous phase through a path relinking template. Computational experiments are reported on 0-1 multiconstraint knapsack problems.

Keywords: integer programming, local search, path relinking, primal-dual interior point method

1. Introduction

Originally designed for solving linear programming problems, interior point methods cover a very wide field including quadratic programming, integer programming, and more recently, semi-definite programming.

Interior point algorithms represent a good alternative to the simplex method, particularly for large problems with an appropriate structured constraint matrix. Consequently, a number of methods using interior point techniques have been developed for integer programming (Mitchell, Pardalos and Resende, 1998).

A recent approach is the combination of interior point methods with metaheuristics. The only existing method, to our knowledge, associates a primal-dual interior point method and a genetic algorithm with an economic cut generator (Schaal, M’Silti and Tolla, 1997). It is based on constraint satisfiability techniques in order to verify a significant number of constraints in a short computational time. The main interest of this method is its ability quickly to find a range of feasible solutions within...
the context of multi-criteria analysis. The interior code comes into play for generating a starting point on which the genetic algorithm will be applied.

First, this technique has inspired us to use an interior code in order to build not only one point but a population of points on which a metaheuristic process will work. Then, to go beyond a simple linking of an interior point method and a metaheuristic, we propose the introduction of the information given by a path relinking (or scatter search) phase into the interior search. Thus, the hybrid method that we have implemented for solving a 0-1 linear program is an original combination of a primal-dual interior point method, an economic cutting plane, and a metaheuristic. It starts with a hybrid search, composed of an interior code and economic cuts, which builds a population of diversified solutions on which the metaheuristic will be applied.

Section 2 defines notations and the problem statement. In Section 3, after describing our hybrid search, called RLPI, which combines a primal-dual interior point method with cutting planes, we report the computational experiments on the 0-1 multiconstraint knapsack problem (MCKP). A combination of RLPI procedure and path relinking search comprises our hybrid method devoted to 0-1 programs. It is studied in Section 4, which details other improved computational experiments on 0-1 MCKP. The last section gives our conclusions and perspectives.

2. Notations and problem statement

The following integer linear program (P) is considered throughout the paper:

\[
\begin{align*}
\text{maximize} & \quad z = f(x) \\
\text{s.t.} & \quad Ax \leq b \quad (P) \\
& \quad x \in \mathbb{N}^n,
\end{align*}
\]

where \( A = (a_{ij})_{i,j} \in \mathbb{N}^{m \times n} \) and \( b \in \mathbb{N}^m \)

Given (P), let us note:

\((\bar{P})\): problem (P) when \( x \in \mathbb{R}^n \) is substituted for \( x \in \mathbb{N}^n \).

\( X = \{ x | Ax \leq b, x \in \mathbb{N}^n \} \)

\( \bar{X} = \{ x | Ax \leq b, x \in \mathbb{R}^n \} \)

3. A hybrid search: combination between a primal-dual interior point method and cutting planes

We have developed a hybrid search which finds a good solution of (P) by means of a decomposition of (\( \bar{P} \)) into two sub-problems by the way of adding a constraint (Boucher et al., 1999).

This search is based on information produced by several applications of an interior code to (\( \bar{P} \)) straightened by a constraint issuing from an economic cut. The economic cuts of (\( \bar{X} \)) are generated through ascent heuristics.

The search is stopped when no new cut is found or if a fixed number of iterations is reached. During
the hybrid search, a sequence of diversified feasible integer points is generated. Those points are
memorized to initialize the population on which our evolutionary method will be applied.

The RLPI algorithm

The recursive algorithm of our hybrid search raised above is called RLPI. Its entering argument is an
integer problem \((P_c)\).

More precisely, feasible integer points are computed by \(RLPI(P_c)\), composed of the five steps
described below.

1. LP-solve \((P_c)\) by a primal-dual interior point method with a particular early termination; the result
is a real interior point \(y\);
2. Round \(y\) to the nearest feasible integer point:
   \[\tilde{y} = \text{arr}(y)\];
3. Apply \(p\) ascent heuristics to \(\tilde{y}\) in order to obtain \(p\) locally optimal integer solutions:
   \[x^i = \text{ASC}_i(\tilde{y}) \quad i = 1, \ldots, p\];
4. Among the solutions found above, choose a solution having the best economic value:
   \[x^e = \arg \max_{i=1,\ldots,p} \{f(x^i)\}\];
5. If \(x^e\) has already been calculated in a previous step then STOP
else
   – construction of two sub-problems:
     \((P_1) \equiv (P|f(x) \geq f(x^e))\) and \((P_2) \equiv (P|f(x) \leq f(x^e))\);
     – \(RLPI(P_1), RLPI(P_2)\).

The initial parameter \((P_c)\) of RLPI is the original problem \((P)\).
The hybrid search stops when no new cuts are added; in particular, a branch is pruned if the solution
value computed at the node has been already encountered in a previous node.

At the end of the local search, we only keep the best encountered solution.
The algorithm, as it is built, corresponds to a search tree, the root of which is \((P)\):

\[
(P) \quad \begin{array}{c}
\downarrow \\
(P_1) \equiv (P|f(x) \leq f(x^e)) \\
\downarrow \\
(P_1|f(x) \leq f(x^{e,1}))
\end{array} \quad \begin{array}{c}
\downarrow \\
(P_2) \equiv (P|f(x) \geq f(x^e)) \\
\downarrow \\
(P_2|f(x) \geq f(x^{e,1}))
\end{array}
\]

Figures 1 and 2 illustrate the different algorithm steps at some RLPI call and show how several
feasible integer points are computed in one stage with a feasible interior point method.
Step 1 of the algorithm consists in LP-solving \((P)\) by the early termination unfeasible ‘predictor-corrector’ primal-dual interior point method (Mehrotra, 1992).

Concerning the first call to \(RLPI\), the termination is standard. By standard we mean the stopping criterion used in the interior point code of the linear programming software CPLEX 6.6 version. This criterion is a duality gap of less than a \(10^{-6}\) threshold. This termination gives an optimal solution (with the required accuracy) of \(P_c\).

For the next calls, we use our own stopping criterion: the interior progression is stopped when the current interior point satisfies the matrix constraints within \(10^{-6}\), i.e. \(|A_i x - b_i| \leq 10^{-6}\) for all \(i = 1, \ldots, m\). Consequently, the iteration number of the interior point method employed is considerably decreased. Nevertheless, the fractional interior point obtained is strictly in the interior of the continuous domain \((\bar{D})\) but remains close to its frontiers. To move the point far from the boundaries, we have

\[
\begin{align*}
(P_1) & = (P | f(x) \geq f(x^*)), \\
(P_2) & = (P | f(x) \leq f(x^*))
\end{align*}
\]
employed the technique of adding an economic cut having the form \( f(x) \leq f(x^e) \) (which leads to a dual degeneracy) and then applying an early termination primal-dual interior point method to the created sub-problem. As a result, the fractional interior point obtained is more centred; this makes it easier to generate feasible integer points.

In order to apply the method, we have chosen to work with 0-1 linear programs. In this case, the rounding in Step 2 is classically computed through the following rule:
arr: $\mathbb{R}^n \rightarrow \{0, 1\}^n$

$y \rightarrow \tilde{y}:\begin{cases} 
\tilde{y}_i = 1 & \text{if } y_i \geq 0.5 \\
\tilde{y}_i = 0 & \text{otherwise}
\end{cases}$

If the 0-1 vector found above is unfeasible then a standard greedy repairing procedure is applied to it.

Given $f(x) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$, the greedy heuristic is composed of two phases:

1. Sort the variables according to the ratio $r_j = c_j/\sum_{i=1}^m u_ia_{ij}$ for $j = 0, \ldots, n$ when $\{u_1, \ldots, u_m\}$ is a set of surrogate (Glover, 1968) multipliers.
   To obtain reasonably good multipliers, we make use of the values of dual variables (Pirkul, 1987) given by the LP-solving of $(P_c)$ by a primal-dual interior point method.
2. Test each variable according to the increasing order of the $r_j$ ratios and complementing variables equal to 1 if feasibility is violated.

Two kinds of improving procedures have been tested:

a) Step 3 is performed through a couple of ascent methods $ASC_1$ and $ASC_2$ (we fix $p$ equal to 2) applied to the integer solution obtained in the previous step.
   The first ascent method is a bit modification (switch) and the other is a bit-exchange (swap).
   For instance, from a solution vector $x = (0, 1, 1, 0)^t$, the neighborhood $N(x)$ generated by a switch or a swap procedure is respectively
   $- N_{\text{switch}}(x)$:
   $x'1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad x'2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad x'3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad x'4 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

   $- N_{\text{swap}}(x)$ (among the six neighbors of $x$ only 4 are different from $x$):
   $x'1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad x'2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad x'3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad x'4 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$

   The main idea of this method is the search of an improved solution among all the vectors, different from $x$, having the same number of ‘0’ and ‘1’.

   The solution $x^{*}$ selected in $N(x)$ is the one which has the best objective evaluation (i.e. $f(x^{*}) \geq f(x') \forall x' \in N(x)$).

   These standard ascent methods stop when they find a local optimum.

b) Step 3 is the opposite of the repairing procedure in Step 2, i.e., it examines in a greedy way each variable of the solution found in the previous step in the decreasing order of the $r_j$ ratios (described in Step 2) and changes the current variable from 0 to 1 while feasibility is not violated.

As we have explained above, $RLPI$ builds a sequence of feasible integer solutions. The best solution is updated at each call to $RLPI$. 
The best solution provided by our local search is often close to optimality or to the best known bound. This assertion is supported by experimental results presented in the following subsection.

**Experimental results**

Tests were made on a Pentium III workstation (500 MHz, 256 Mo RAM) using SUSE LINUX 6.4. The interior point method used is implemented by CPLEX 6.6 and our heuristic is coded in C. To apply our hybrid search, we have chosen the well-known 0-1 multiconstraint knapsack problem, denoted in the sequel **MCKP**.

The **MCKP** has the following form:

\[
\begin{align*}
\text{maximize} \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i \in \{1, \ldots, m\} \quad (\text{MCKP}) \\
\end{align*}
\]

\[x_j = 0 \text{ or } 1 \quad j \in \{1, \ldots, n\}\]

assuming that all the data are integer and non-negative.

This problem is relevant for testing our method, which includes *path relinking* and diversification/intensification procedures, since it has been widely treated and particularly lots of metaheuristics have been applied to it: simulated annealing (Drexl, 1988), tabu search (Glover and Kochenberger, 1996; Hanafi and Fréville, 1998), genetic algorithms (Thiel and Voß, 1994; Chu and Beasley, 1998), threshold accepting algorithm (Dueck and Wirsching, 1991).

In a first version of **RLPI**, Step 3 is performed through the couple of ascent methods **ASC1** and **ASC2** described above.

Experimentations have been carried out on 56 instances of the **MCKP**, with the number of variables varying from 27 to 105 and the number of constraints from 2 to 30. Those problems were collected by Fréville and Plateau (1994) and are available in the **OR library** (http://mscmga.ms.ic.ac.uk/jeb/orlib/). Since exact solutions are known for these instances, they represent an interesting base of comparison for easily measuring the quality (in terms of relative gap to optimality) of the best found solution.

For the 56 problems base, successive **RLPI** calls until the ending criterion is satisfied allow us to solve 50% of the instances. For 90% of problems, the relative gap to optimality is lower than 1%. The average gap to optimality is equal to 0.45%.

The gap to optimality is computed through the following formula:

\[
\frac{\text{optimal value} - \text{best value computed by RLPI}}{\text{optimal value}}
\]

The second set of problems used for testing our local search is the Chu and Beasley database (Chu and Beasley, 1998). Those instances of the **MCKP** were generated by Chu and Beasley using the procedure suggested by Fréville and Plateau (1994). The number \(m\) of constraints is equal to 5, 10, or 30 and the number \(n\) of variables is respectively set to 100, 250, or 500. Thirty problems are generated for each combination of \(m\) and \(n\): the database comprises 270 problems.
Each set of thirty instances is built according to the following rule:

a) the right-hand side coefficients \( b_i \ (i \in \{1, \ldots, m\} \) are computed by

\[
b_i = \alpha \sum_{j=1}^{n} a_{ij}
\]

where

- \( \alpha \) is a tightness ratio:

\[
\alpha = \begin{cases} 
0.25 & \text{for the first ten instances} \\
0.5 & \text{for the next ten} \\
0.75 & \text{for the last ten}
\end{cases}
\]

- \( a_{ij} \) are integer numbers drawn from the discrete uniform law \( U(0, 1000) \).

b) The coefficients \( c_j \ (j \in \{1, \ldots, n\} \) of the objective function are correlated to the \( a_{ij} \) through the formula

\[
c_j = \sum_{i=1}^{m} a_{ij}/m + 500q_j \quad j \in \{1, \ldots, n\}
\]

where \( q_j \) is a real number drawn from the continuous uniform generator \( U(0, 1) \).

Correlated problems are generally more difficult to solve (Pirkul, 1987). Moreover, this base is harder than the previous one because optimal solutions are not all known and the sizes of the instances are significantly larger. The best solutions, currently known, for this base were computed by a genetic algorithm designed by Chu and Beasley (Chu and Beasley, 1998). They have been recently improved by Hao and Vasquez (2000). Tested databases are both available on the OR library. Table 1 contains the preliminary computational experiments.

The first two columns in Table 1 indicate the size \( (m \cdot n) \) and the tightness ratio \( (\alpha) \) of a particular instance structure. The third column reports the average gap to the best solution found by Chu and Beasley over 10 problem instances for each problem structure:

<table>
<thead>
<tr>
<th>best value obtained by Chu and Beasley - best value computed by RLPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>best value obtained by Chu and Beasley</td>
</tr>
</tbody>
</table>

The last column indicates the average execution time (in CPU min:sec) of RLPI.

For the most difficult instances (in the table, from the set of instances having 10 constraints and 500 variables), we have only reported results with Step 3 performed by a bit modification procedure. Indeed, the swap procedure in Step 3 improves the solution quality for some instances but increases the computational time (in CPU minutes:seconds): Table 2 sums up the (average) results.

In Table 1, the results relative to the best lower bound found by Chu and Beasley are quite satisfying: 24% are lower than a gap of 0.1%, and 85% lower than a gap of 1%. We observe the same characteristics as those of the results reported by Chu and Beasley. First, the tightness ratio \( \alpha \) comes into play concerning the quality of the gap to the best value found by Chu and Beasley. Indeed, logically, the smaller \( \alpha \) is (i.e., the tighter constraint is), the larger the gap is. Second, for the same \( m \) and \( \alpha \), as \( n \) increases, the problems become much harder, take more time to be solved, and the gap to the best value found by Chu and Beasley is better.
If \( m \) increases while \( n \) and \( \alpha \) are fixed, we have the same observations as the previous case concerning the increasing difficulty. Nevertheless, the average gap grows with the constraint number. As a matter of fact, the instances with 500 variables and \( \alpha \) equal to 0.75 provide the best results. There also exists a link between the structure of the instances and their LP-values. Thus, the best results obtained concern sets of instances whose average gaps to the LP-value ((LP-value – best value found by Chu and Beasley)/LP-value) are the smallest in the paper by Chu and Beasley. So, RLPI has more chance to find a better gap while the optimal integer value is close to the LP-value of the problem.

A modified RLPI procedure: insertion of new cuts and computation of analytic centers

We recall that the swap procedure in Step 3 improves the solution quality but increases the computational time (in CPU seconds) for the largest problems. This result has encouraged us to only
use a bit modification neighborhood in the improving heuristic and to perform Step 3 through the greedy heuristic sorting by surrogate multipliers. On the other side, in order to keep the same best solution quality, we have added refinements. We explain them in this section. Experimental results devoted to the 0-1 MCKP associated with this refined method are reported after having combined it with a path relinking process. This combination is proposed through our hybrid method in the next section.

**Insertion of new cuts**

A way for tightening the continuous domain is to add new cuts. The added cuts take advantage of the information given by RLPI. Indeed, the cutting planes consist in excluding the unfeasible 0-1 points \( \tilde{y} \) generated during the rounding procedure. Cuts are computed as follows (Maculan, 2001):

\[
\forall x = (x_1, \ldots, x_n) \in \{0, 1\}^n \quad \sum_{i=1}^{n} |x_i - \tilde{y}_i| \geq 1
\]

Consequently, this cut can be written as the following simple linear constraint:

\[
\sum_{i: y_i=0} x_i + \sum_{i: y_i=1} (1 - x_i) \geq 1
\]

Figure 3 is a graphic illustration of how the unfeasible point \( \tilde{y} \) is excluded.

**Computation of analytic centers**

Concerning the computation of strictly interior points, instead of LP-solving \((P)\), we set to 0 all the coefficients of the objective function. This technique allows us to generate a pure analytic center.
Indeed, by cancelling the coefficients of the objective function, the original problem, once modified by a barrier function in the interior point method, has the following form:

\[
\begin{align*}
\text{maximize} & \quad z = \sum_{j=1}^{n} \ln(x_j) = \ln(\prod_{j=1}^{n} x_j) \\
\text{s.t.} & \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i \in \{1, \ldots, m\} \\
& \quad x_j \in [0, 1], \quad j \in \{1, \ldots, n\}
\end{align*}
\]

Maximizing the product of the \( x_j \) amounts to maximizing the product of the distances to the frontiers of the continuous domain. This is the reason why solving this modified problem provides an analytic center of the continuous domain associated with \((P)\).

4. Hybrid method

The fairly good results given by \textit{RLPI} have comforted us in using it for starting a hybrid method including metaheuristics for 0-1 programs. The main idea consists in combining our local search \textit{RLPI} with \textit{path relinking}. Indeed, an interior point method in a metaheuristic scheme can provide a diversified initial population.

In our local search \textit{RLPI}, we only update the best solution computed. Nevertheless, all the solutions \( x^e \) encountered through the search process form a diversified population. Thus, a way to build the initial population of a metaheuristic consists in storing all the points generated during the tree search. This technique enables us to have a larger set of solutions so that the \textit{path relinking} phase begins in good
conditions. Indeed, the larger the population is, the more extended the exploration of the continuous domain will be. In this section, we call this version of RLPI ‘modified RLPI’.

Before detailing our experimental results, we explain how the classical path relinking heuristic is integrated in our method.

**Path relinking and RLPI**

**Some generalities about path relinking**

Before getting to the heart of the combination, let us briefly describe the basic principles of path relinking.

The scatter search and its generalized form called path relinking (Glover, 1995, 1998) are evolutionary methods that have recently been shown to yield promising outcomes for solving combinatorial and non-linear optimization problems.

As usual in this type of heuristics, we need to construct an initial population. Here, this population is created by using the 0-1 points encountered in the modified RLPI scheme described earlier. Scatter search is described intensively in Glover (1995, 1998). The main characteristics consist in performing linear combinations of elite solutions and using a rounding process in the integer case.

We choose path relinking, which considers a combination of solutions in the solution neighborhood space. This method consists in generating paths between and beyond the elite solutions, where solutions on such paths also serve as sources for generating additional paths. In fact, a path is composed of 0-1 solutions. The endpoints of a path are called initiating solution (starting point) and guiding solution (final point). The intermediate solutions of the path, which happen to be unfeasible, are calculated in order to draw near to the structure of the guiding solution according to the Hamming distance (the Hamming distance between two 0-1 vectors is the number of different components), subsequently called $d_H$.

The path relinking is applied to the population built by our modified RLPI through the following phases:

- selection of a pair of solutions in the population;
- choice of the initiating and guiding solutions;
- computation of the neighborhood;
- selection of a solution in the neighborhood.

The different steps in the path construction are detailed in the next section.

**Path relinking algorithm**

Let us denote by $x^*$, the best encountered solution.

- $x^1$ and $x^2$, two solutions in the population,
- $x^0$, the initiating solution and $x^g$, the guiding one,
- $S$, the index set of different variables between $x^0$ and $x^g$ ($|S| = d_H(x^0, x^g)$, this set decreases by one element at each iteration)
- $S_0$ (resp. $S_1$) $S$, the index set of variables equal to 0 (resp. equal to 1) in $x^0$.

The function of path construction between $x^1$ and $x^2$ called $PATH(x^1, x^2)$ is composed of the following steps:
Initiating phase:

\[
\text{If } f(x^1) < f(x^2) \text{ then } x^0 \leftarrow x^1; x^g \leftarrow x^2
\]
\[
\text{else } x^0 \leftarrow x^2; x^g \leftarrow x^1
\]
\[
S \leftarrow \{ j \in \{1, \ldots, n\} : x^0_j \neq x^g_j \}
\]
\[
S_0 \leftarrow \{ j \in S : x^0_j = 0 \}; \ S_1 \leftarrow \{ j \in S : x^0_j = 1 \}
\]
\[
x^* \leftarrow x^g
\]

While \( S \neq \emptyset \) do

Build the neighborhood \( N(x^0) \) of \( x^0 \): \( N(x^0) = \{ x : d_H(x, x^0) = 1 \} \);

Pick the solution \( x' \in N(x^0) \) defined by:

\[
x'_i = \begin{cases} 
1 - x^0_i & \text{if } i = k \\
x^0_i & \text{if } i \neq k 
\end{cases}
\]

where \( k \in S \) is defined by

\[
k = \begin{cases} 
\arg \min \left\{ \sum_{i=1}^{m} c_j : j \in S_1 \right\} & \text{if every } x' \in N(x^0) \text{ satisfying } \sum_{i} x'_i = 1 + \sum_{i} x^0_i \\
\arg \max \left\{ \sum_{i=1}^{m} u_i a_{ij} : j \in S_0 \right\} & \text{otherwise}
\end{cases}
\]

if \( f(x') > f(x^*) \) then \( x^* \leftarrow x' \)
\[
x^0 \leftarrow x' ; S = S \setminus \{ k \}
\]

endwhile;

output(\( x^* \))

This function returns the best encountered solution \( x^* \) in the path.

We denote \( P = \{ x^1, \ldots, x^p \} \), the population built by the modified RLPI.

At the end of this procedure, if \( x^* \) is better than the worth solution \( x^- = \arg \min_{x \in P} f(x) \) of the population, then it replaces it.

This procedure is applied to all pairs of solutions in the population. The whole path relinking algorithm has the form:

While the stopping criterion is not satisfied do

for each \((x^i, x^j) \in P^2, i < j, \) not examined yet do

- \( x^* = PATH(x^i, x^j) \)
- if \( f(x^*) > f(x^-) \) then

\[
P \leftarrow (P \setminus \{ x^- \}) \cup \{ x^* \}
\]

endwhile

The stopping criterion is a fixed number of iterations for which the best solution computed is not improved.
Experimental results of the modified RLPI

Table 3 gives computational results after refinements and the path relinking process. As in Table 1, we compute an average gap between the best solution computed by Chu and Beasley and the best one provided by our own heuristic. Gaps are better than in Table 1, and our average execution time decreases to 14.62 seconds. The Chu and Beasley genetic algorithm provides an average execution time of 1267.4 seconds on a Silicon Graphics Indigo workstation (R4, 100 MHz, 48 MB main memory) using CPLEX solver version 4.0. In this table, the third column is the number of new cuts added.

<table>
<thead>
<tr>
<th>Problems m.n</th>
<th>α ratio</th>
<th>Average # cuts</th>
<th>Average gap</th>
<th>Average CPU time (m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.100</td>
<td>0.25</td>
<td>2.9</td>
<td>0.0190</td>
<td>00:01.20</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>14.4</td>
<td>0.0033</td>
<td>00:01.89</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>47.0</td>
<td>0.0018</td>
<td>00:04.27</td>
</tr>
<tr>
<td>5.250</td>
<td>0.25</td>
<td>1.5</td>
<td>0.0093</td>
<td>00:02.66</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>17.6</td>
<td>0.0013</td>
<td>00:04.72</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>48.8</td>
<td>0.0007</td>
<td>00:10.02</td>
</tr>
<tr>
<td>5.500</td>
<td>0.25</td>
<td>1.8</td>
<td>0.0027</td>
<td>00:05.08</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>18.4</td>
<td>0.0008</td>
<td>00:10.70</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>46.5</td>
<td>0.0003</td>
<td>00:21.05</td>
</tr>
<tr>
<td>10.100</td>
<td>0.25</td>
<td>1.2</td>
<td>0.0147</td>
<td>00:01.56</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>17.7</td>
<td>0.0038</td>
<td>00:02.83</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>54.0</td>
<td>0.0026</td>
<td>00:05.72</td>
</tr>
<tr>
<td>10.250</td>
<td>0.25</td>
<td>2.9</td>
<td>0.0068</td>
<td>00:04.32</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>18.8</td>
<td>0.0025</td>
<td>00:07.50</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>52.5</td>
<td>0.0010</td>
<td>00:13.77</td>
</tr>
<tr>
<td>10.500</td>
<td>0.25</td>
<td>16.9</td>
<td>0.0038</td>
<td>00:12.38</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>21.7</td>
<td>0.0014</td>
<td>00:17.04</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>54.3</td>
<td>0.0005</td>
<td>00:31.07</td>
</tr>
<tr>
<td>30.100</td>
<td>0.25</td>
<td>0.7</td>
<td>0.0272</td>
<td>00:04.76</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>18.7</td>
<td>0.0040</td>
<td>00:07.56</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>61.8</td>
<td>0.0033</td>
<td>00:13.11</td>
</tr>
<tr>
<td>30.250</td>
<td>0.25</td>
<td>1.1</td>
<td>0.0124</td>
<td>00:11.33</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>22.6</td>
<td>0.0031</td>
<td>00:18.62</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>60.3</td>
<td>0.0017</td>
<td>00:32.88</td>
</tr>
<tr>
<td>30.500</td>
<td>0.25</td>
<td>15.6</td>
<td>0.0050</td>
<td>00:32.21</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>22.3</td>
<td>0.0016</td>
<td>00:44.19</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>60.0</td>
<td>0.0009</td>
<td>01:12.43</td>
</tr>
<tr>
<td>Average</td>
<td>26.0</td>
<td>0.0047</td>
<td></td>
<td>00:14.62</td>
</tr>
</tbody>
</table>

Source: Chu and Beasley (1998)
5. Conclusions

Our local search RLPI, hybridizing an interior point process with ascent methods, provides a sample of distinct solutions in the sense of the Hamming distance. Those are an attractive base for building an initial population of an elaborate local search (metaheuristic).

The computational experiments of RLPI and a path relinking scheme give a promising way for a directed exploration of the feasible domain. Particularly, using interior point methods for integer programming allows us to compute diversified fractional interior points in different parts of the constraint polyhedron. Moreover, the good results found by our method applied to MCKP convince us to introduce refinements concerning the use of improved directions given by several points of the population into the interior code process.

Moreover, our perspective consist in implementing the framework of a new hybrid method illustrated by Fig. 4 which will include diversification and intensification phases using interior directions computed by a primal-dual interior point method.

Validation tests are in progress. However, the former results look quite attractive.

Acknowledgement

The authors wish to express special thanks to Nelson Maculan, who provided the idea of cuts eliminating the unfeasible integer points found after a rounding procedure.

Fig. 4. The hybrid method.
References


