INFORMS Journal on Computing
Vol. 19, No. 1, Winter 2007, pp. 73–79
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1901 | 0073

DOI 10.1287/ijoc.1050.0146

Linear Complementarity as a General Solution Method to Combinatorial Problems

Laura Di Giacomo, Giacomo Patrizi
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università "La Sapienza," 00185 Rome, Italy
{lauradg@caspur.it, g.patrizi@caspur.it}

Emanuele Argento
Dipartimento di Informatica Università "La Sapienza," 00185 Rome, Italy, emanuele@banach.sta.uniroma1.it

This paper shows how many types of combinatorial problems can be embedded in continuous space and solved as nonconvex optimization problems. If the objective function and the constraints are linear, problems of this kind can be formulated as linear complementarity problems. An algorithm is presented to solve this type of problem and indicate its convergence properties. Computational comparisons are carried out using general solution codes.

Key words: programming complementarity integer-programming applications
History: Accepted by John N. Hooker, Jr., Area Editor for Logic, Optimization, and Constraint Programming; received February 2000; revised February 2002, September 2004, March 2005; accepted April 2005.

1. Introduction

We propose an algorithm to solve some important combinatorial problems with a linear objective function, subject to integer or binary variables over a polyhedral set (Mitchell and Todd 1992). The solution method for this restricted class is formulated by embedding the problem in an appropriate space, so that at the solution, the variables will only assume the permitted integer values. The advantage of this method is that all the results of mathematical analysis can be used to obtain the solution.

Many approaches have been tried to solve combinatorial problems (Nemhauser and Wolsey 1988): enumeration techniques, implicit enumeration methods, branch-and-bound approaches, cutting planes, relaxation, and heuristic methods (Rinooy Kan 1986, Hochbaum 1997).

Direct nonlinear methods, such as the one to be described in this paper, have also been formulated. Here variables are constrained to allow only integer values, giving rise to a nonlinear nonconvex constrained problem, which must be solved (Giannessi and Nicolucci 1976).

In the next section a characterization of combinatorial problems will be given, together with the transformations adopted and the solution method. In §3 some computational results will be presented for diverse classes of problems, while in §4 the relevant conclusions will be drawn.

2. The Solution Method

Combinatorial problems are defined over a discrete structure, rendering the methods of mathematical analysis difficult to apply. The aim of this section is to show how many such problems can be transformed into continuous problems and present a solution algorithm.

DEFINITION 1 (MITCHELL AND TODD 1992). A linear combinatorial optimization problem is a combinatorial optimization problem, limited to variables that can take on only binary values $(0, 1)$ defined over a polyhedral constraint set and selected through a linear objective function.

Let $c \in \mathbf{R}^n$ and $b \in \mathbf{R}^m$ be vectors and C an $m \times n$ matrix. Then the linear combinatorial optimization problem may be defined as:

$$\begin{gathered} \text{Max}[c^T x:\ Cx + b \geq 0] \\ x \in \{0, 1\}^n \end{gathered} \tag{1}$$

For all linear combinatorial problems the objective function is bounded because there are finitely many feasible points.

Further, let M be an $n \times n$ matrix, let $q \in \mathbf{R}^n$ be a given vector, and let $x \in \mathbf{R}^n$ be the solution vector to be determined. Then the linear complementarity problem (LCP) is:

$$\begin{gathered} Mx + q \geq 0 \\ x \geq 0 \\ x^T(Mx + q) = 0 \end{gathered} \tag{2}$$

This problem is a unifying formulation for many problems in optimization and operations research, such as quadratic programming, bimatrix games, market equilibrium, optimal capital stock, and optimal stopping problems (Cottle et al. 1992). Under suitable

conditions, it is in fact a statement of the Kuhn-Tucker necessary conditions for optimality for a quadratic program.

Any linear combinatorial problem can be embedded in a linear complementarity problem. If the objective function exists, it can be added as a constraint and the solutions must be recursively obtained by altering the value of the affine term of this constraint, through a dichotomous search on the parameter (Nemhauser and Wolsey 1988).

Let e be a vector of ones of appropriate dimension, Γ another nonnegative vector of appropriate dimension, which will be null throughout, and μ a nonnegative scalar, which will also be null throughout. Then any linear combinatorial optimization problem can be represented by an LCP in the following form:

$$\begin{pmatrix} -I & 0 & 0 \\ C & 0 & 0 \\ c^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \Gamma \\ \mu \end{pmatrix} + \begin{pmatrix} e \\ b \\ -k \end{pmatrix} \geq 0 \tag{3}$$

$$\begin{pmatrix} x \\ \Gamma \\ \mu \end{pmatrix} \geq 0 \tag{4}$$

$$(x^T \quad \Gamma^T \quad \mu)^T \cdot \left(\begin{pmatrix} -I & 0 & 0 \\ C & 0 & 0 \\ c^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \Gamma \\ \mu \end{pmatrix} + \begin{pmatrix} e \\ b \\ -k \end{pmatrix} \right) = 0 \tag{5}$$

and the following result holds.

THEOREM 1. *The linear combinatorial optimization problem has an optimal solution with a value of the objective function equal to k^*, if the linear complementarity problem has a solution with the given parameter $k = k^*$ and the variables Γ and μ are equal to zero. Conversely, let the linear complementarity problem have a solution with Γ and μ equal to zero and a given value of $k = \hat{k}$. This will be a solution to the linear combinatorial optimization problem if there is no $k > \hat{k}$ that solves the linear complementarity problem.*

PROOF. Let the linear combinatorial optimization problem have an optimal solution with a value of the objective function k^* and solution vector x^*. If the parameter k is set at $k = k^*$ the solution is feasible for the linear complementarity problem since it obviously satisfies (3) and (4). Moreover, the solution satisfies (5), since for the given value of the solution vector all elements satisfy the condition $x_i(1 - x_i) = 0$, while the other two vectors, being null, also ensure satisfaction of the equation.

For the converse, assume that the linear complementarity problem has a solution. Then by the complementarity condition (5) we have $x_i(1 - x_i) = 0$. The only values that will satisfy this equation are $x_i = 0, 1$, and the solution forms a binary-valued vector. Moreover the solution vector ensures that the constraints of the linear combinatorial problem (1) are satisfied. The objective function $c^T x$ has an upper bound, so by a simple dichotomous search procedure the least value of $-k$ that will still render the problem feasible can be determined, while any decrease in $-k$ will render the linear complementarity problem infeasible. Finally throughout Γ, and μ are null. This provides the optimal solution to the linear combinatorial optimization problem.

COROLLARY 1. *If the cost elements of the objective function have integer values then the optimal solution can be found by solving at most $\log_2(Z)$ linear complementarity problems, where Z is the maximum value of the objective function.*

The following algorithm has been proposed for solving any LCP (Patrizi 1991). The LCP problem is transformed into a linear program with a parametric variation of a scalar parameter. Thus consider the LCP (2) and embed the problem so as to form a new coefficient matrix and affine vector with:

$$a_i = d_i - \min[0, C_{ij}, -q_i : C_{ij} \in C_{i.}]$$
$$i, j = 1, 2, \ldots, n,\ i \neq j\ d_i \geq 0\ i = 1, 2, \ldots, n$$

given by:

$$A = \begin{pmatrix} 1 & e^T \\ a & C + ae^T \end{pmatrix}, \qquad s = \begin{pmatrix} -\alpha \\ q - a\alpha \end{pmatrix}, \qquad c = p^T A$$

where α is an upper bound to the sum of the elements in the solution, and p is the largest positive eigenvector of the positive matrix A. Note by this transformation the matrix A is rendered all positive, while the affine vector s is all negative. By a theorem of Frobenius, it is assured that such an eigenvalue exists (Berman and Plemmons 1979).

Let $i \in I$ be an index set of basic variables to the linear-programming (LP) problem and let $j \in J$ be an element belonging to the index set of variables that are non basic. In obvious notation the essential result is the following:

THEOREM 2 (PATRIZI 1991). *A nontrivial and feasible LCP has a solution x^* such that the sum of the solution elements is equal to or less than a given parameter α, i.e. $\alpha \geq e^T x^*$, if and only if the parametric LP problem with a scalar parameter c_o, given by:*

$$c^T u + c^T z + c_o u_o = \text{Min } Z$$
$$Au + (I - A)z + au_o + s \geq 0$$
$$e^T u + e^T z + u_o - \alpha = 0$$
$$u_o, u, z \geq 0$$

has a dual nondegenerate solution for some c_o with all the elements of the vector z non basic in the primal solution, and with $A_{jI}A_{II}^{-1} \geq 0$ for some $j \in J$. From a primal optimal solution $(u^*, 0, u_o^*)$ to the parametric LP problem above, which satisfies the conditions, a solution to the original LCP (2) is obtained immediately by letting $x^* = u^*$.

Thus to solve an LCP, an LP is defined, as indicated in the theorem, and the solution is determined, with a parametric variation on the scalar variable c_0. If a solution results that is nondegenerate in the dual and if the condition on the coefficients is satisfied, then by Theorem 1 it is a solution to the combinatorial problem.

Computationally, the combinatorial problems may be ill-conditioned and the transformation of the linear combinatorial problem may lead to an LCP with many rows that may be linearly dependent. The conditions of the theorem may, therefore, be difficult to satisfy with sufficient accuracy given the specific finite precision of the LP routine used.

So a branch-and-bound procedure has been devised to enforce the conditions in such cases, given that the problem has a solution.

Thus consider the noncomplementary solution to the problem obtained after a full parametric variation of the coefficient c_0 and choose the feasible solution that exhibits the least violation in the complementarity condition. The first variable that appears in the feasible solution without the integrality condition is set to one, while the associated variables in the constraints are set to zero. This will result in one fixed variable and a new iteration is carried out with the full parametric variation on c_0. Either a complementary solution will result, or once more a feasible non complementary solution is found, which will cause the process to be repeated for a second variable, and so on until, if necessary, all the combinations of these non integer variables have been verified, as in a normal branch-and-bound process.

This procedure is obviously finite, and it need only be applied to the fixed variables that receive nonintegral values due to finite precision or dual degeneracy. It is used, for instance in the problems that are logical contradictions, which accounts for the large computing times in Table 1.

Table 1 Selected Results of the Solution of Satisfiability Problems

Name	Literals	Clauses	GLPK	XPRESS	FORTMP	LCP
ulm027r0	25	64	5.00	2.00	3.00	0.53
ulm054r1	46	128	6.00	4.00	3.00	2.72
ulm081r1	92	256	5.00	3.00	2.00	4.12
ulm189r1	161	448	7.00	4.00	4.00	145.49
ulmbc040	40	340	6.00	3.00	4.00	2.96
ulmbp160#	160	485	*	4.00	63.00	5,762.73
ulmbs180	180	540	*	5.00	18.00	9,558.82
real2b12	15	111	6.00	4.00	2.00	0.21
real2l12	15	119	6.00	4.00	2.00	0.17
real2q11	16	100	5.00	4.00	3.00	0.24
real2x11	14	77	6.00	4.00	3.00	0.15
hole8#	72	297	150.00	4.00	27.00	518.06
hole9#	90	415	*	6.00	193.00	5,770.87
hole10#	110	561	*	3.00	301.00	19,505.87

Note. Time is in seconds.
*Solution not feasible.
#Instance is logical contradiction.

3. Computational Results

Problems of three classes have been solved with this procedure and the computational results are presented in §§3.1, 3.2, and 3.3.

This LCP approach to solve combinatorial problems of the type indicated is a general technique and therefore should be compared to other general techniques to solve combinatorial problems, rather than specialized heuristics. Thus the comparisons proposed serve to indicate the computational difficulties in the problems attempted.

It should be noticed, however, that the benchmark algorithms are production codes for various branch-and-bound implementations, while this LCP code is a preliminary experimental code that is in no way optimized. Thus the significance of this research is to provide a comparison of the feasibility of this approach compared to the well-known branch-and-bound methodology.

The codes used for comparison were taken from the Neos Server (http://www-neos.mcs.anl.gov/neos/) during 2002–2003 and included:

- GLPK, a package formerly available on the Neos server and is now available for download. The package is intended for solving large-scale LP, mixed-integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.
- XPRESS-MP (version 12.13) is a commercial product of Dash Optimization (http://www.dashoptimization.com/). It uses a branch-and-bound algorithm to solve the problems. As there may be an exponential number of possible solutions, the package reduces the number of solutions to a manageable size by applying various cutting planes. A presolver is also used to tighten the formulation, which should improve the quality of initial solutions and enables better cutting planes to be generated.
- FortMP 3.2e is a mathematical-programming system designed by N.A.G. (National Algorithmic Group, http://www.nag.co.uk/) and others to solve large scale LP, integer-programming (IP) and quadratic mixed-integer programming (QMIP) models by solving LP models, namely, sparse simplex (SSX) algorithms comprising PRIMAL and DUAL, and interior-point method (IPM) algorithms based on the primal-dual logarithmic (predictor-corrector) barrier method.
- CPLEX (version 8.0) is a commercial product of Ilog (http://www.ilog.com/products/cplex/) very similar to XPRESS-MP driven by the operating system Windows 2000. The machine is an HP 920 at 181.851 Mflops on the basis of the Dongarra Benchmark test suite. The solution method to be applied to solve these problems was determined automatically, as CPLEX has an evaluation facility to determine if a presolver should be called or cutting planes should be used or when branch and bound should be applied. This pack-

age was used only for the third set of experiments, since for the other tests the package was not available.

While these provide some of the most popular routines available, the Neos server timing facility was, during the experimentation period, not very accurate in providing an estimate of the time required to find a solution. The times reported are based on the time required to unpack the package, compile the code, and solve the problem.

This provides an order-of-magnitude estimate of solution times, which can be very useful to evaluate the feasibility of this LCP approach for combinatorial problems, as comparisons can be made with other benchmark test problems on the Neos server and on the preferred machine and solver, so that the relative times may be evaluated. The computational results reported should be sufficiently extensive and in line with the aim of the paper. Further comparisons are left to the interested reader.

The computational times for the CPLEX experimentation are indicated with the solution results and were converted, for comparison purposes, through the Dongarra benchmark suite to equivalent times on a Digital 4100 workstation with two processors at 600 MHZ running OSF1 4.0, which was the machine on which the LCP routine to solve the emergency-service problem was run.

3.1. Satisfiability Problems in Normal Conjunctive Form

A satisfiability problem in conjunctive normal form is represented as the intersection of a set of m clauses, where each clause is the union of a set of up to n literals, which may receive the value of true or false (1 for true and 0 for false). Representing these literals as boolean variables, each literal can be indicated in affirmative form by x_i $i = 1, 2, \ldots, n$ or in the negated form as $1 - x_i$. Thus, without loss of generality, any clause may be represented with the index set of affirmative literals I and the index set of negated literals J. The clause is true if

$$\sum_{i \in I} x_i + \sum_{j \in J} (1 - x_j) \geq 1.$$

As all the clauses must be true for the expression to be true, the satisfiability problem may be rewritten in terms of the real number system as:

$$Cx + b \geq 0$$
$$x_i \in \{0, 1\}$$
$$-1 \leq b_j \leq n - 1, \quad j = 1, \ldots, m.$$

This can be written as an LCP with coefficient matrix and affine vector

$$M = \begin{pmatrix} -I & 0 \\ C & 0 \end{pmatrix} \qquad q = \begin{pmatrix} e \\ b \end{pmatrix}.$$

This algorithm has been applied to more than 94 satisfiability problems (Mayer et al. 1995) and gave excellent results on the problems attempted and a selection of the results is shown in Table 1.

The three procedures used for comparison are general branch-and-bound linear LP algorithms. The algorithm presented here compares favorably to the results obtained. No routine or special selection procedures were enacted to take advantage of the special nature of the problem. The calculations were performed on IBM 6000 240 and PowerPC machines and the comparison is between different general procedures described above. The time estimates quoted are the actual times reported by the solvers.

Better results could be obtainable if the specialized structure of the problem were exploited, but this would make the algorithms no longer general procedures.

The first four problems were recursively generated by the addition of binary equalities, to a set of base problems, while the next three problems are particular combinatorial problems, such as determining how many queens can be placed on an $n \times n$ chess board so that no queen can capture another. The next set of four are real problems arising from VLSI tests and thus deal with practical applications. Finally the last three hole problems are unsolvable logical problems (Hintikka 1955).

The solution found for each problem was checked that it was a binary solution and that it was feasible for the complete problem.

3.2. Knapsack Problems

The knapsack problem maximizes the value of the items carried in a knapsack, subject to a capacity constraint, which must always be active to make the problem interesting.

Any knapsack problem may be expressed as a combinatorial problem in the following way:

$$\text{Max}[c^T x: a^T x - b \leq 0]$$
$$x \in \{0, 1\}^n$$

It is a simple problem containing an objective function and a single constraint.

The objective function must be transformed into an inequality constraint, so that a two-row matrix results:

$$\begin{pmatrix} -a^T \\ c^T \end{pmatrix} (x) + \begin{pmatrix} b \\ -k \end{pmatrix} \geq 0$$

The coefficient k must be increased from iteration to iteration, by dichotomous search, as indicated in §2.

Thus the LCP formulation of any knapsack problem is:

$$\begin{pmatrix} -I & 0 & 0 \\ -a^T & 0 & 0 \\ c^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \gamma \\ \mu \end{pmatrix} + \begin{pmatrix} e \\ b \\ -k \end{pmatrix} \geq 0$$

$$\begin{pmatrix} x \\ \gamma \\ \mu \end{pmatrix} \geq 0$$

$$(x^T \quad \gamma \quad \mu)^T \left(\begin{pmatrix} -I & 0 & 0 \\ -a^T & 0 & 0 \\ c^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \gamma \\ \mu \end{pmatrix} + \begin{pmatrix} e \\ b \\ -k \end{pmatrix} \right) = 0$$

where the notation is as before and the variables γ, μ are scalars that remain at their zero value throughout the solution procedure, as is evident.

The LCP routine was implemented and compared to the three alternative solvers, as indicated above. The test set was constructed by considering five instances per group. Each instance in a group was formulated by considering a specific probability distribution with differing parameters to generate the coefficients of the instances to be solved.

Instances of various sizes were generated by sampling the value of an object and its weight from the probability distributions that were considered. Use was made of the **R** statistical package (Dalgaard 2002) for the distributions (uniform and geometric).

- Uniform distribution: the weights and values for these instances were drawn from a uniform distribution on $[0, n]$ where n is the number of items. Five instances for every size class were generated.
- Geometric distribution: the geometric distribution depends on a parameter $0 < p < 1$, and has a mass function given by $f_n = p(1-p)^{n-1}$. The mean of the distribution for a given p is $\mu = 1/p$ while the variance of the distribution is $\mu_2 = (1-p)/p^2$ (Kendall and Stuart 1958). Thus for low values of the parameter p the geometric distribution is similar to a uniform distribution, while for increasing values of the parameter p it becomes more and more peaked and bunched so that the probability of selecting identical values or ones that are arbitrarily close increases greatly. Here, this will imply that many objects have similar weights and values. The test set was generated for values from $p = 0.4$ through 0.9, and five instances were generated for each value.
- Mixed distributions: the same procedure as above was carried out for the weights and for the values of the objects, by generating the data in exactly the same way from the appropriate distribution.

As can be seen from Table 2, in many cases, the branch-and-bound procedures are very fast and accurate in determining a solution. This is especially true for the problems generated from the uniform distribution, but the solution times for the problems become progressively worse as the weights and the values of the objects are bunched together. This is obtained by using different probability distributions with smaller variances.

Thus, in general the solution times for these randomly generated problems depend on the parameter and the type of probability distribution used and not only on the size of the problem.

Table 2 Average Solution Results of Five Knapsack Problems with Different Routines and Distributional Assumptions

Distribution	Size	GLPK	XPRESS	FORTMP	LCP
Uniform	100	6.00	5.00	3.40	23.53
	200	6.25	5.25	4.75	64.64
	500	9.50	3.50	3.50	390.17
Geometric $p = 0.4$	100	222.30	6.20	7.20	22.24
Geometric $p = 0.5$	100	7,320.00	5.00	12.33	24.18
Geometric $p = 0.6$	100	%	744.20	11.20	22.82
Geometric $p = 0.8$	100	%	379.20	%	25.41
	200	%	240.00	%	85.09
Geometric $p = 0.9$	100	%	3,607.00	%	24.80
	200	%	945.00	%	72.06
Geom/uniform $p = 0.4$	500	%	3.00	18.0	310.37
Uniform/geom $p = 0.4$	500	%	3.00	10.0	332.17

Note. Time is in seconds.
*No termination before 1,200 secs.

On the other hand, with the LCP routine, an LP is solved and the parametric variation of the scalar parameter is carried out, so the computational times will depend on the size of the problem, which is what is evident in the table. The distributional characteristics have hardly any influence on the solution times with this algorithm and the small variations in the computational times may be due to the parametric variation rather than the lack of dual feasibility.

3.3. The Emergency-Service Problem

The emergency-service problem consists of distributing optimally over a territory a limited number of services in relation to the possible demand so as to ensure a given service level. Typically, the ambulance emergency-service problem determines the optimal location of ambulances over a territory (Mercuri 1992).

Usually the locations consider two types of points: service points, where the ambulances just wait for a service call, and facility points such as hospitals, where ambulances deliver the patients and, often for want of better planning, await the next call.

The IP problem is:

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{b_{it}} Q(p_t, q_t, (k-1))(1-q_t) q_t^{k-1} a_{it} y_{ikt} = \text{Min } Z \quad (6)$$

$$\sum_{j \in M_{it}} x_{jt} + \sum_{k=1}^{b_{it}} y_{ikt} \geq b_{it} \quad (7)$$

$$\sum_{j \in J} x_{jt} \leq p_t \quad (8)$$

$$x_{jt} - c_{ht} z_{ht} \leq 0 \quad (9)$$

$$\sum_{h \in J'} z_{ht} \leq l_t \quad (10)$$

$$x_{jt} \in integer$$

$$y_{ikt} \in \{0, 1\}$$

$$z_{ht} \in \{0, 1\}$$

The index sets I, J, J' and T are respectively the index sets of demand nodes, service nodes, facility nodes, and the time period considered. The integer and binary variables to be determined are: $y_{ikt} = 1$ if demand node $i \in I$ is not covered by k services at time t; $z_{ht} = 1$ if a facility node is covered at $h \in J'$ at time t; x_{jt} is the number of emergency services assigned to node $j \in J \cup J'$ at time t.

Thus, let a_{it} be the number of service requests at demand node i at time t, q_t be the probability that a service unit is busy at time t and b_{it} be the number of service units required to cover node i at time t with probability β that will depend on the request pattern and the desired probability level of service (Mercuri 1992). Then (6) attempts to minimize the probability of not covering the emergency requests with a service coming from a node in the set of service or facility points given by M_{it}, which is an index set of service and facility points that cover demand point i at time t, within the required distance.

The set of constraints (7) requires that for every demand node i at each time $t \in T$ the number of services available plus the services that are missing is greater than or equal to the number of services necessary to ensure that the requests are covered with at least the given probability of service. The total number of services available at all service nodes must be less than the number available at time t, i.e., p_t (8). The number of emergency units stationed at a facility node $h \in J'$ must not be greater than its capacity c_{ht} (9), while all possible facility nodes should be less than or equal to a given number for every time period $t \in T$ (10). Finally, the variables are either integer or binary.

The LCP version of this IP was applied to two well-known problems (Mirchandani 1985, Torregas et al. 1971) and to a number of large applications regarding Florence and Rome. The integer variables required were defined by a suitable number of binary variables, through the binary expansion of the integers. This is a very straightforward way to transform a combinatorial problem defined over the integers to a combinatorial problem defined over binary variables and thus suitable for this LCP formulation without altering the algorithm in any way.

The structure and the computational results for each problem are presented in Tables 3 and 4. In the Table 3 the first column has the name of the problem instance and in the next three columns its composition is specified regarding the number of demand points of a service point and the number of time periods considered. Column five has the number of vehicles present per period. For the six-period Mirchandani instance, either two or three vehicles were available at each period, for a total of 16 vehicles over the interval, or an average number of vehicles per period of 2.67. Similar considerations apply to the next problem, while for the set of problems regarding the Italian cities the number of vehicles was kept constant throughout the day.

Table 3 Solution of the Emergency-Vehicle Problem: Many Emergency Vehicles

Name	Demand pts.	Service pts.	Periods	Vehicles	Cplex#	LCP
Mirchandani	10	10	6	2.67*	7.77	0.77
Torregas	30	30	3	7.67*	5.55	62.96
Florence1	30	30	6	40	8.88	585.40
Florence2	80	80	2	40	11.00	397.95
Florence3	95	95	2	40	13.20	742.39
Rome1	100	100	2	40	16.5	242.77
Rome2	120	120	1	40	8.8	457.49
Rome3	220	220	1	40	33.0	1,825.16
Rome4	150	150	1	40	16.5	507.94
Rome5	200	200	1	40	27.50	2,229.12

Note. Time is in seconds.
#Machine equivalent times (Dongarra) ratio cplex/Lcp = 110.
*Average of emergency vehicles available over the periods.

In Table 3 the number of vehicles is relatively large, in line with the actual number of ambulances available in the municipalities. As the vehicles are abundant, complete covering was available in most cases.

Computational solution times obtained for the CPLEX routine have been multiplied by 110, which is the ratio of the performance efficiency of the two machines, (Dongarra 2004). The particular calculation was carried out with LinpackJava (www.netlib.org/benchmark/linpackjava).

Apart from the first instance, which is a fairly small problem, all the other instances run much faster with CPLEX. Moreover, the relative difference between computational times increases with increasing size of the problem, which indicates a much better handling facility for large data sets in CPLEX compared to the LCP.

In Table 4, completely different computational results appear, for the case in which the emergency vehicles are relatively scarce. In this set of problems only the Italian-city instances are considered and the number of emergency vehicles is reduced to eight vehicles per period. The results for the Italian-city instances

Table 4 Solution of the Emergency-Vehicle Problem: Scarce Emergency Vehicles

Name	Demand pts.	Service pts.	Periods	Vehicles	Cplex#	LCP
Florence1	30	30	6	8	30.8	318.91
Florence2	80	80	2	8	21,567.70	319.91
Florence3	95	95	2	8	..*	678.26
Rome1	100	100	2	8	116.60	761.91
Rome2	120	120	1	8	..*	350.33
Rome3	220	220	1	8	..*	1,891.96
Rome4	150	150	1	8	..*	488.14
Rome5	200	200	1	8	..*	1,443.40

Note. Time is in seconds.
#Machine equivalent times (Dongarra) ratio cplex/Lcp = 110.
*Not solved in 3,000 secs of actual machine time.

in Tables 3 and 4 should be compared. The computational times for the instances that can be solved with CPLEX in Table 4, considering an upper computational time limit of 50 minutes (3,000 seconds) increase many-fold compared to those in Table 3. Instead for the LCP routine, the tighter constraints due to a smaller number of emergency vehicles tend to reduce the computational times.

Exactly the same LP must be solved for the LCP routine in both cases, with a different right-hand side in some inequalities. It is obvious that computational times will be very similar, except for the computation of the parametric variation, which appears to be reduced for the more heavily constrained problems.

Where CPLEX was unsuccessful within the time limits imposed, there seems to be a critical ratio of the number of service points per emergency vehicle per period, which seems to be around ten. If this ratio is above ten then the number of branch-and-bound iterations becomes excessive, except in the Rome1 instance.

Once again the results are very encouraging and show that effectively large locational problems can be solved by the LCP algorithm, as in all the other problem types considered.

4. Conclusions

The method adopted provides reliable and reasonably fast results to the solution of combinatorial problems, which can be applied to any linear combinatorial optimization problem with binary variables. Since parametric linear programming is recognized as an NP-complete algorithm (Murty 1972) no polynomial results are claimed for this algorithm, which is also apparent from considering the computational results.

Due to its generality, the LCP algorithm can be tried as a first approach to solving a given problem, whereupon a specialized procedure can be constructed if desired.

It is the only solver that consistently solves all the problem instances considered. Its computational times are relatively invariant with respect to the structure of the problem and depend only on the number of variables and constraints present in the instance. This is often important in experimental work, where changes in the structure of the coefficient matrix may intervene. While these modifications may have a marked effect on the solution times of the branch-and-bound procedure, they will affect only marginally those of the LCP procedure.

Further, given a new instance of a combinatorial problem in a given number of variables the estimate of the time taken to solve the problem with a branch-and-bound procedure is likely to be very uncertain, while with the LCP routine this is likely to be much less so.

The algorithm applied should be demonstrably convergent (either a solution is specified, or the fact that the problem does not have a solution is indicated), within a reasonable time span (Curry 1963). Further, the chosen algorithm should be semantically adequate, which implies that it should solve the problem or indicate that no solution exists, for all variations in the structure of the coefficient matrix or of the right-hand side.

From the limited computational results provided here, it appears that the LCP algorithm satisfies these criteria to a greater extent than do the other branch-and-bound procedures, but much more experimentation is required on this score.

Acknowledgments

The authors thank Prof. John Hooker for all his attention and efforts and an anonymous referee for all his or her helpful comments. Needless to say, the authors are responsible for any lingering mistakes.

References

Berman, A., R. J. Plemmons. 1979. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, New York.

Cottle, R. E., J.-S. Pang, R. E. Stone. 1992. *The Linear Complementarity Problem.* Academic Press, New York.

Curry, H. B. 1963. *Foundations of Mathematical Logic.* McGraw-Hill, New York.

Dalgaard, P. 2002. *Introductory Statistics with R.* Springer-Verlag, Berlin, Germany.

Dongarra, J. J. 2004. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301; Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 27831.

Giannessi, F., F. Nicolucci. 1976. Connections between nonlinear and integer programming problems. Istituto Nazionale di Alta Matematica. *Symposia Mathematica,* Vol. XIX. Academic Press, New York.

Hintikka, K. J. J. 1955. Form and content in quantification theory. *Acta Philosophica Fennica* **8** 7–55.

Hochbaum, D. S. 1997. *Approximation Algorithms for NP-Hard Problems.* International PWS Publ. Co., Boston, MA.

Kendall, M., A. Stuart. 1958. *The Advanced Theory of Statistics,* Vol. 1. Griffin, London, UK.

Mayer, J., I. Mittereiter, F. J. Radermacher. 1995. Running time experiments on some algorithms for solving propositional satisfiability problems. *Ann. Oper. Res.* **55** 139–178.

Mercuri, M. 1992. Soluzioni di problemi di localizzazione dei servizi di emergenza in situazioni aleatorie e dinamiche. Tesi di Laurea, Facoltà di Scienze Statistiche, Demografiche ed Attuariali, Universitá di Roma "La Sapienza," Rome, Italy.

Mirchandani, P. B. 1985. The M median problem. *Eur. J. Oper. Res.* **21** 121–137.

Mitchell, J. E., M. J. Todd. 1992. Solving combinatorial problems using Karmarkar's algorithm. *Math. Programming* **56** 245–284.

Murty, K. J. 1972. On the number of solutions to the complementarity problem and spanning properties of complementary cones. *Linear Algebra Its Appl.* **5** 65–108.

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and Combinatorial Optimization.* Wiley, New York.

Patrizi, G. 1991. The equivalence of an LCP to a parametric linear program with a scalar parameter. *Eur. J. Oper. Res.* **51** 367–386.

Rinnooy Kan, A. H. G. 1986. An introduction to the analysis of approximation algorithms. *Discrete Appl. Math.* **14** 111–134.

Torregas, C., R. Swain, C. Revelle, L. Bergman. 1971. The location of emergency service facilities. *Oper. Res.* **19** 1363–1373.