Lagrangian Techniques for Solving a Class of Zero-One Integer Linear Programs*

Yao-Jen Chang and Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois
Urbana, IL, 61801
{chang,wah}@manip.crhc.uiuc.edu

Abstract

We consider a class of zero-one integer programming feasibility problems (0-1 ILPF problems) in which the coefficients of variables can be integers, and the objective is to find an assignment of binary variables so that all constraints are satisfied. We propose a Lagrangian formulation in the continuous space and develop a gradient search in this space. By using two counteracting forces, one performing gradient search in the primal space (of the original variables) and the other in the dual space (of the Lagrangian variables), we show that our search algorithm does not get trapped in local minima and reaches equilibrium only when a feasible assignment to the original problem is found. We present experimental results comparing our method with backtracking and local search (based on random restarts). Our results show that 0-1 ILPF problems of reasonable sizes can be solved by an order of magnitude faster than existing methods.

1 Introduction

In this paper, we study the search of a feasible assignment to a *zero-one integer linear programming problem* (0-1 ILP) that has binary variables and integral coefficients. We called such problems *zero-one integer linear programming feasibility problems* (0-1 ILPF problems).

A 0-1 ILPF problem with n variables constrained by m equalities is defined as follows. Given A, an m-by-n integer matrix, and b, an m-tuple of integers, does there exist an n-tuple x such that

$$Ax = b \quad \text{where } x \text{ is a binary vector of } n \text{ elements} \tag{1}$$

An alternative form that specifies the integrality constraints explicitly is as follows.

$$\begin{aligned} Ax &= b & (2)\\ {x_i}^2 &= x_i, \quad i = 1, 2, \cdots, n. & (3) \end{aligned}$$

0-1 ILPF problems are NP-hard in general, and solving satisfiability problems can be shown to be equivalent to solving 0-1 ILPF problems with binary coefficients. Methods to solve these problems can be classified into two categories: systematic enumeration of solutions in discrete space, and finding approximate solutions in continuous space.

Search methods in discrete space are generally based on random restarts and methods to evaluate a current (partial) assignment. Examples of these evaluation methods include probabilistic measures (as in simulated annealing), heuristic functions (as in fitness functions of genetic algorithms), and number of constraints violated (as in constraint-satisfaction algorithms). The problem with random restarts is that a search in a seemingly good direction may get stuck in a very small local minimum, and a random restart may bring the search to a completely different search space.

Obviously, we can consider 0-1 ILPF problems as *constraint-satisfaction problems* (CSP). Existing solution methods to solve CFPs include backtracking [9, 4], best-first search, most-constrained first search [13], and local-search methods such as hill-climbing [11, 9, 4]. In practice, these methods have been equipped with heuristic guidance such as conflict minimization [9, 12] to make them more efficient.

Gu [5] proposed a transformation of a discrete satisfiability problem into an unconstrained global optimization problem in real space. Using a combination

*This research was supported by National Science Foundation Grant MIP 92-18715.

Proc. Computer Software and Applications Conference, Aug. 1995.

of gradient descents in continuous space and backtracking in discrete space, Gu showed significant improvements in solution time for solving certain classes of conjunctive normal-form formulae. For other problems, a local search often leads to local minima, and the number of times that the algorithm backtracks grows exponentially with respect to the problem size.

We propose a new search method based on Lagrange multipliers and a transformation of Eq's 2 and 3 into equations in continuous space. We propose a new approach that allows a search to escape from a local minimum and to continue in the same trajectory without restarting from a new starting point. To accomplish this, we use two counteracting forces, one performing local continuous search in the primal space and the other in the dual space. When gradient descent brings the search to a local minimum optimizing Eq's 2 and 3, a counteracting force will be present to bring the search out of the local minimum if the solution obtained is not valid. The advantage of our approach is that it does not rely on random restarts to bring the search out of a local minimum. Rather, the search continues to evolve until a satisfiable solution is found. This avoids the problem of restarting to a new starting point even when a good solution is in the proximity of a local minimum.

2 Lagrangian Formulation of 0-1 ILPF Problems

Lagrangian techniques are well known relaxation methods for solving constrained optimization problems. Little work has been done in applying Lagrangian methods to solve constrained combinatorial search problems [3, 6, 2].

One possible formulation in continuous space of a 0-1 ILPF problem, **CM**, is as follows.

$$\begin{array}{ll} \min & f(x) = \|Ax-b\|_2^2 + \sum_{i=1}^{n}(x_i{}^2 - x_i)^2 \\ \text{s. t.} & Ax = b \\ & x_i{}^2 = x_i, \quad i = 1, 2, \cdots, n. \end{array} \tag{4}$$

In this formulation, we have added an objective function aiming to minimize the least square error of the equality constraints. The objective function will be 0 if all the constraints are satisfied. Obviously, all feasible solutions to Eq. 4 are global and also solve the original ILPF problem.

An alternative formulation of the previous 0-1 ILPF problem in continuous space, **UM**, is as follows.

$$\min f(x) = \|Ax-b\|_2^2 + \sum_{i=1}^{n}(x_i{}^2 - x_i)^2 \tag{5}$$

This unconstrained version gets a feasible solution to the ILPF problem provided that the objective function can be minimized to 0. Unfortunately, this function can have many dent-like local optima with fractional values, which provide no answer to the original problem.

Using Lagrange multipliers, we can formulate the necessary conditions for optimality of Eq. 4 as follows.

$$L(x,\lambda,\mu) = c\left[\|Ax-b\|_2^2 + \sum_{i=1}^{n}(x_i{}^2 - x_i)^2\right] + \lambda(Ax-b) + \sum_{i=1}^{n}\mu_i(x_i{}^2 - x_i) \tag{6}$$

where $c > 0$ is a parameter, and $\lambda \in R^m$ and $\mu \in R^n$ are Lagrange multipliers.

Denote

$$\nabla_x L(x,\lambda,\mu) = \left[\frac{\partial L(x,\lambda,\mu)}{\partial x_1}, \frac{\partial L(x,\lambda,\mu)}{\partial x_2}, \ldots, \frac{\partial L(x,\lambda,\mu)}{\partial x_n}\right]^T \tag{7}$$

$$\nabla_{xx}^2 L(x,\lambda,\mu) = \left[\frac{\partial^2 L(x,\lambda,\mu)}{\partial x_i \partial x_j}\right] \tag{8}$$

We know from classical optimization theory the following fact.

[Fact] x^* is a local minimum for Eq. 4 if and only if x^* is feasible and there exists a unique vector $(\lambda^*, \mu^*) \in R^{m+n}$ such that

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0 \tag{9}$$

and $\nabla_{xx}^2 L(x^*, \lambda^*, \mu^*)$ is positive semi-definite at x^*. ■

Since local optima in our formulation in Eq. 4 are also global, Eq. 9 above suffices as a guide in finding a solution to the ILPF problem defined in Eq. 1. In particular, to solve Eq. 4 by Lagrangian methods, we set up the following first-order necessary conditions, **ALP**:

$$\nabla_x L(x,\lambda,\mu) = 0 \tag{10}$$

$$\nabla_\lambda L(x,\lambda,\mu) = 0 \tag{11}$$

$$\nabla_\mu L(x,\lambda,\mu) = 0 \tag{12}$$

This is is a system of $2n+m$ nonlinear equations with $n+m$ unknowns. We formulate Eq's 10-12 into a dynamic system of differential equations and apply either a finite difference-equation solver or a differential-equation solver to find numerical solutions. The dynamic system can be stated as follows.

$$\frac{dx}{dt} = -\nabla_x L(x, \lambda, \mu) \quad (13)$$

$$\frac{d\lambda}{dt} = \nabla_\lambda L(x, \lambda, \mu) = Ax - b \quad (14)$$

$$\frac{d\mu}{dt} = \nabla_\mu L(x, \lambda, \mu) = \left[{x_1}^2 - x_1,\ {x_2}^2 - x_2, \cdots,\ {x_n}^2 - x_n\right]^T \quad (15)$$

Note that this is a dynamic system that evolves over time t and performs *gradient descent* for primal variables (variables that appear in the original constrained problems) and *gradient ascent* for dual variables (Lagrange multipliers). When an optimal solution to Eq. 4 is found, we arrive at a saddle point in Eq's 13-15.

[Definition] For a Lagrangian function $L(x, \lambda, \mu)$, the *saddle-point condition* is satisfied at (x^*, λ^*, μ^*), if

$$L(x^*, \lambda, \mu) \le L(x^*, \lambda^*, \mu^*) \le L(x, \lambda^*, \mu^*). \quad (16)$$

■

[Saddle-Point Theorem] Arriving at a saddle point is a sufficient condition for optimality. [8] ■

Similar approaches have been used to solve linear and nonlinear programs [1, 14].

3 Illustrative Examples

The example problem, an ILPF problem with 30 variables, has a special structure in such a way that each variable z appears only once, while there are multiple occurrences of the x variables (the z's are slack variables). Although it is not a difficult problem to solve, it illustrates how our proposed Lagrangian method works.

[Example 1] Find a binary solution $(x_1, \cdots, x_6, z_1, \cdots, z_{24})$ such that

$$\begin{aligned}
-x_1 + x_2 + x_4 - z_1 - z_2 &= 0 \\
x_1 - x_2 + x_4 - z_3 - z_4 &= 0 \\
1 + x_2 - x_3 - x_6 - z_5 - z_6 &= 0 \\
1 - x_1 - x_2 + x_4 - z_7 - z_8 &= 0 \\
2 - x_4 - x_5 - x_6 - z_{10} - z_9 &= 0 \\
x_1 - x_4 + x_6 - z_{11} - z_{12} &= 0 \\
x_1 - x_5 + x_6 - z_{13} - z_{14} &= 0 \\
-x_1 + x_2 + x_6 - z_{15} - z_{16} &= 0 \\
-x_1 + x_3 + x_6 - z_{17} - z_{18} &= 0 \\
1 - x_4 - x_5 + x_6 - z_{19} - z_{20} &= 0 \\
1 + x_2 - x_4 - x_5 - z_{21} - z_{22} &= 0 \\
1 - x_2 - x_4 + x_6 - z_{23} - z_{24} &= 0
\end{aligned}$$

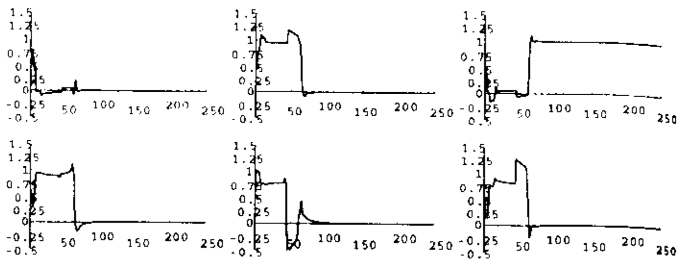

Figure 1: The transient behavior of $\vec{x} = (x_1, x_2, \cdots, x_6)$ in Example 1 when c is set to 5. The horizontal axis is in time constants, which are logical times in the differential equation solver. The CPU time it takes to solve this example is 241 seconds.

The given ILPF problem is first transformed using Eq. 4 into a continuous dynamic system using Lagrange multipliers. We then apply *Livermore Solver for Ordinary Differential Equations* (LSODE)[1], a popular package for solving differential equations.

Using $(\vec{x}, \vec{z}) = (\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})$ as a starting point (for reason of symmetry) and setting $c = 5$, the dynamic system defined by Eq's 13-15 gives a trajectory as shown in Figure 1. The trajectory shows drastic transitions, with each variable reaching near-zero and near-one points before settling at a stable and satisfiable equilibrium at $\vec{x} = (0, 0, 1, 0, 0, 0)$.

4 Experimental Results

In this section, we show more extensive results on experimenting with our proposed method.

4.1 Test-Case Generation

The test cases we are interested in have between 120-210 variables and 50-90 constraints. We did not

[1] *Livermore Solver for Ordinary Differential Equations* (LSODE) is the basic solver of *ODEPACK* [7] developed in Lawrence Livermore National Laboratory.

use small test cases as they can be solved easily by depth-first search or by gradient-descent methods with random restarts.

The way that test cases were generated is as follows.

- Only equality constraints were generated.
- The left-hand side of an equality has five variables, each multiplied by a coefficient from the set {1, 2, −1, or −2}. We have selected coefficients of the same order of magnitude to eliminate the dominance of solution by coefficients of significantly greater magnitude. By doing so, we have created test problems that are more difficult to solve.
- The right-hand side of an equality can be any integer from the set {0, −1, 1, −2, 2}.
- Each variable and coefficient is randomly selected from its possible set with equal probability.

4.2 Algorithms Tested and Experimental Conditions

We have applied three algorithms to solve the test cases generated.

- **Proposed Lagrange-Multiplier Method.** For consistency, all dual variables were set to be 0 and all primal variables to be 0.5 as starting points. The convexity term c was set to be 5.
- **Depth-First Search (Backtracking)**[2] . For comparison purposes, backtracking is used as a constraint-based search algorithm. Using a sparse matrix dual simplex algorithm for linear relaxation, depth-first backtracking can prune partially solved subproblems as soon as these subproblems are determined as infeasible.
- **Local Searches with Random Restarts.** Since local-search methods can only be applied to an unconstrained objective function, we use Eq. 5 as the objective function. We applied Newton's gradient descent method[3] and restarted the algorithm when a local minimum is reached. When restarting a search, each variable was set to be 0 or 1 with equal probability.

[2] The package we have used was developed by Michel at Eindhoven University of Technology in the Netherlands. It is a very efficient mixed integer linear problem solver that is widely used to solve large scale LP/ILP/MILP problems on scalar and vector processors.

[3] This package was written by S. G. Nash [10] at George Mason University and is available on the Internet.

In our experiments, we have chosen to terminate an algorithm when it has run for a preset limit over another algorithm that has found a feasible solution.

- We ran local search with random restarts using the same amount of time taken by the Lagrangian method.
- If the Lagrangian method can solve a test case within 3 hours, then we ran the depth-first search for a maximum of 12 hours before terminating it.
- If the Lagrangian method cannot determine whether a test case is feasible or not within 3 hours, then we ran the backtracking algorithm for a maximum of 3 hours.
  - If the depth-first search gets a solution in 3 hours, then we ran the Lagrangian method for another 9 hours so that the total run time of the Lagrangian method is also 12 hours.
  - If the backtracking algorithm could not find a solution in 3 hours, then we terminate the search and discard the test case.

We ran all our experiments on single-processor Sun SparcStation 20s.

4.3 Results

Table 1 summarizes the results we have obtained in our experiments. Column 1 is the size of the test cases characterized by m and n, where n is the number of variables and m the number of constraints. Columns 2, 4, and 7 (*Succ*) are labeled by "Y," "N," or "?" that indicate, respectively, the test case is feasible, not feasible, and undetermined. For the Lagrangian and local-search methods, *Succ* can only be "Y" or "?" as they will not terminate if the test case is not feasible. *Succ* for the depth-first search can be "Y," "N," or "?." When *Succ* is "N," infeasibility had been proved and all nodes expanded were pruned. When labeled by "?," the algorithm was terminated due to excessive CPU time. We have also shown in Column 5 the number of nodes expanded by depth-first search, an indicator of how difficult a test case is. We have the following observations on each individual method.

- The local-search method can find feasible solutions easily when test cases are small or when there are few constraints. That is, the method works well when the search space is small or when there are sufficiently large numbers of feasible solutions.

Table 1: Performance of the three methods in solving sample ILPF problems.

	Lagrangian		Backtracking			Local
ILPF ID	*Succ*	CPU time	*Succ*	# nodes expanded	CPU time	*Succ*
n=130	Y	52:16	Y	11840	23:10	?
m=55	Y	28:52	?	>232887	>8:26:07	?
	Y	13:57	Y	8976	26:33	?
	?	1:18:50	?	>202459	>10:18:56	?
	Y	12:03	Y	77	10	?
n=140	Y	17:31	Y	2257	6:34	?
m=55	Y	21:48	Y	18408	54:29	?
	Y	14:26	Y	977	2:48	?
	Y	12:33	Y	2158	6:40	?
	Y	39:47	?	>228205	>10:18:06	?
n=150	Y	27:16	Y	2111	7:23	?
m=55	Y	20:13	?	>257463	>10:18:34	Y
	Y	19:30	Y	67	8	Y
	Y	17:24	Y	86	11	Y
	Y	15:59	Y	13169	34:31	Y
n=140	?	1:18:01	Y	133386	3:54:43	?
m=60	Y	41:23	Y	232110	7:44:54	?
	Y	24:52	Y	1822	3:20	?
	?	1:22:56	?	>544071	>13:10:40	?
	?	1:08:04	N	34259	1:09:00	?
n=150	Y	13:43	Y	144927	4:26:41	?
m=60	Y	20:30	Y	325	33	?
	Y	9:47	Y	4662	9:32	?
	Y	11:27	Y	17278	31:49	Y
	Y	29:37	Y	148274	4:12:15	?
n=160	Y	22:25	?	>181394	>5:25:10	Y
m=60	Y	28:21	?	>454071	>13:00:04	Y
	Y	25:30	?	>311296	>10:22:20	Y
	Y	18:43	Y	77	6	?
	Y	24:02	Y	38694	1:02:26	?
n=150	Y	48:41	Y	73071	2:40:59	?
m=65	Y	1:25:48	?	>325856	>10:13:00	?
	Y	17:08	Y	162928	6:31:46	?
	?	2:05:45	?	>264126	>13:01:00	?
	Y	1:00:30	?	>176713	>9:19:48	?
n=160	Y	46:09	Y	82799	3:35:22	?
m=65	Y	50:10	?	>119369	>10:18:19	?
	Y	48:20	Y	1228	6:29	?
	Y	2:23:55	Y	2136	9:09	?
	Y	49:02	?	>121390	>9:19:48	?
n=170	Y	24:49	?	>266666	>11:35:22	?
m=65	Y	45:16	?	>289060	>10:58:21	?
	Y	30:09	?	>243419	>11:24:55	Y
	Y	38:02	Y	12851	30:46	Y
	Y	20:09	?	>279698	>11:49:50	?
n=170	Y	1:41:58	?	>135763	>8:54:42	?
m=70	Y	1:43:18	?	>83090	>8:24:22	?
	Y	1:01:09	?	>97133	>8:18:17	?
	Y	1:59:02	?	>73728	>8:17:19	?
	Y	48:29	Y	54929	4:26:31	?
n=190	Y	53:20	?	>135753	>8:20:00	?
m=80	Y	1:03:00	?	> 83090	>12:58:39	?
	Y	1:56:11	?	> 97133	>10:10:55	?
	Y	2:23:57	?	> 73728	>10:25:20	?
	?	2:51:52	?	> 196090	>13:03:15	?
n=210	Y	2:36:16	?	>108836	>12:02:35	?
m=90	Y	1:33:20	?	> 86601	>10:33:13	?
	Y	2:05:12	?	>120111	>11:01:01	?
	?	3:03:11	?	> 98454	>12:22:03	?
	Y	2:11:07	?	> 97133	>11:45:16	?

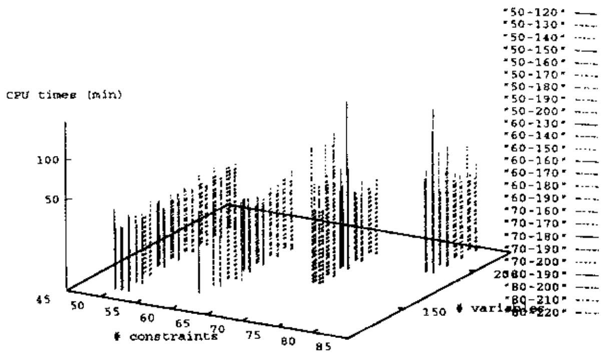

Figure 2: Problem sizes versus run-times of the Lagrangian method for solving randomly generated ILPF problems based on the method described in Section 4.1.

- Our proposed Lagrangian method works best when the test case is feasible but the number of solutions is small. In this case, its complexity is related to the number of feasible solutions in the solution space and not so much on the size of the solution space. This phenomenon is illustrated in Table 1 that shows little variations in run times for test cases of the same size and that shows relatively small increase in complexity when the problem size is increased.
- The depth-first backtracking algorithm is better than the Lagrangian method when test cases are small or when test cases are easy with few constraints. It is also suitable for proving infeasibility, which cannot be done by local-search methods or the Lagrangian method. However, it is prohibitively expensive for large test cases. Moreover, run times within a group of problems can have large variances, making it difficult to report an average performance.

We have been able to show a speedup of 10 using the proposed Lagrangian method for most of the test cases experimented, especially when test cases are large. As we often cannot finish the depth-first search in the time allowed, the real speedup should be even greater.

Figure 2 shows the relationship between problem sizes and run-times of the Lagrangian method. We have shown run-time results for problems of various numbers of variables and constraints and, for each problem size, raw CPU times of each experiment. Some entries in Figure 2 are left blank due to the difficulty of generating feasible test cases of these problem sizes. These results show that complexity grows moderately for test cases we have studied. Although the results are not conclusive, they illustrate a promising approach for solving problems in this class.

5 Conclusions

In this paper, we have proposed a Lagrangian method for solving 0-1 integer linear feasibility problems with integral coefficients. In our Lagrangian formulation, locally optimal solutions correspond to globally optimal ones; hence, our method is a global optimization method.

The introduction of Lagrangian variables provides a counteractive force that brings the search trajectory out of local optima, even in the presence of many local optima in the original solution space. This method of escaping from a local minimum cannot be achieved in other local-search methods. Our method, however, does not solve infeasible problems. In this case, enumeration-based methods, such as depth-first backtracking, must be used to prove infeasibility.

References

[1] A. Cichocki and R. Unbehauen. Switched-capacitor artificial neural networks for nonlinear optimization with constraints. In *Proceedings of 1990 IEEE International Symposium on Circuits and Systems*, pages 2809–2812, 1990.

[2] B. Gavish and S. L. Hantler. An algorithm for optimal route selection in sna networks. *IEEE Transactions on Communications*, pages 1154–1161, 1983.

[3] A. M. Geoffrion. Lagrangian relaxation and its uses in integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[4] J. Gu. Local search for satisfiability (sat) problem. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(4):1108–1129, 1993.

[5] J. Gu. Global optimization for satisfiability (sat) problem. *IEEE Trans. on Knowledge and Data Engineering*, 6(3):361 381, Jun 1994.

[6] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 6:62–88, 1971.

[7] Alan C. Hindmarsh. odepack, a systematized collection of ode solvers. In R. S. Stepleman et al., editor, *Scientific Computing*, pages 55–64. North-Holland, Amsterdam, 1983.

[8] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1984.

[9] S. Minton, M. D. Johnson, A. B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, Dec 1992.

[10] S. G. Nash. Newton-type minimization via the lanczos method. *SIAM Journal of Numerical Analysis*, 21:770–778, 1984.

[11] Bart Selman and Henry Kautz. Domain-independent extensions to gsat: Solving large structured satisfiability problems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 290–295, 1993.

[12] R. Sosič and J. Gu. Efficient local search with conflict minimization: A case study of the n-queens problem. *IEEE Trans. on Knowledge and Data Engineering*, 6(5):661–668, 1994.

[13] H. S. Stone and J. M. Stone. Efficient search techniques – an empirical study of the n-queens problem. *IBM J. Res. Dev.*, 31:464–474, 1987.

[14] S. Zhang and A. G. Constantinides. Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 39(7):441–452, 1992.