Non-Systematic Backtracking for Mixed Integer Programs

Steven Prestwich
Cork Constraint Computation Centre
University College, Cork, Ireland
s.prestwich@cs.ucc.ie

Armagan Tarim
Artificial Intelligence Group
Department of Computer Science
University of York,York, U.K.
at@cs.york.ac.uk

Abstract

A variety of hybrids of Constraint Programming, Artificial Intelligence and Operations Research techniques have given impressive results. Three recent approaches are (i) the use of relaxations in constraint systems, (ii) non-systematic backtracking to boost the scalability of constraint solvers, and (iii) non-systematic backtracking to boost the scalability of branch-and-bound search. This paper describes a hybrid of all three approaches that combines non-systematic backtracking, SAT-based inference and linear relaxation. It is intended for large MIPs that are hard for reasons of both optimisation and feasibility. Such problems are of practical as well as theoretical interest and we expect the hybrid to find many applications. It is currently under development and results will be reported at the workshop.

Introduction

Mixed integer programs (MIPs) have many practical applications yet are among the hardest problems to solve. The usual approach to such problems is to apply classical Operations Research algorithm such as branch-and-bound, branch-and-cut and branch-and-price (Nemhauser & Wolsey 1988), but local search algorithms have also been used (Brockmann & Decker 2000).

A variety of CP, AI and OR techniques have recently been integrated into hybrid algorithms, sometimes with impressive results. (Gomes 2000) discusses the rich set of connections between Constraint Programming (CP), Artificial Intelligence (AI) and Operational Research (OR), focusing on approaches for dealing with hard combinatorial problems, and highlighted the potential synergistic benefits. This paper describes a new hybrid for MIP that combines several features: the Simplex algorithm to handle real variables, to detect inconsistency earlier, and to guide variable assignment; constraint propagation to locate feasible solutions quickly; and non-systematic backtracking to boost scalability to that of local search. All pairs of these features have previously been combined successfully; though combining good heuristics does not always work well, this leads us to believe that a meta-hybrid of all three will also perform well.

The aim is to solve large MIPs that combine hard optimisation with few feasible solutions. The remainder of this section provides background material.

Propagation and relaxation

Constraint programming systems use constraint propagation to eliminate infeasible variable assignments, and is the tool of choice for combinatorial problems in which feasible solutions are hard to find. Operations Research, on the other hand, uses more sophisticated cost reasoning to prune search spaces, and to locate high-quality solutions. The two disciplines have complementary strengths and weaknesses, and their combination can yield results that are superior to either alone. Many papers have been written in the last few years on this type of hybrid approach, and here we mention just a few.

Linear relaxations and propagation are combined in (Gomes & Shmoys 2002); scalability is improved by restarting the search at intervals, randomised rounding is used as a value ordering heuristic, and relaxation overheads are reduced by applying Simplex only at selected search nodes. Relaxations and propagation are also combined in (Bosch & Trick 2002), and relaxation overheads reduced by computing them only on the most important constraints. (Bokmayr & Kasper 1998) introduces a unifying framework, branch and infer, to describe and compare the languages of integer linear programming and finite domain constraint programming, both from the viewpoint of model building and model solving. Their framework shows how integer linear programming can be extended with symbolic constraints and how algorithmic techniques from integer programming can be used in combination with finite domain methods. Branch-and-check (Thorsteinsson 2001) integrates mixed integer programming and constraint logic programming, encapsulating the traditional Bender's decomposition and branch-and-bound as special cases, and utilising nogood learning. (Caseau & Laburthe 1995) compares propagation to integer programming, and they explain why propagation works better than integer programming on disjunctive problems. (Rodosek, Wallace, & Hajian 1999) presents (i) an automatic transformation of CLP programs, (ii) improvements on such transformations, and (iii) a hybrid algorithm which reduces the solution space of the problem by using finite domain propagation and a dual simplex algorithm. In (Hajian *et al.*

1998) a finite domain solver is used to compute a feasible solution as a starting point for Simplex. In (Do & Kambhampati 2000) LP relaxations are used to detect infeasibility in SAT problems, and to provide seed solutions for SAT solvers. SAT propagation methods are interleaved with an incremental Simplex algorithm in (Wolfman & Weld 1999) to solve SAT problems with metrics. In (Focacci, Lodi, & Milano 1999) non-linear constraints can be used to provide lower cost bounds. A survey of combinations of constraint programming and optimisation techniques is given in (Hooker 2000).

This work shows that constraint propagation and cost reasoning can be profitably combined. Cost reasoning plays the part of a global constraint, guiding the search to useful regions of the search space. Constraint handling performs efficient local reasoning to eliminate infeasible variable assignments.

Propagation and local search

The complementary strengths of constraint programming and local search have also been noted in many papers. Constraint programming uses powerful propagation techniques to prune search spaces, while local search uses constraints in a relatively naive way. Nevertheless, local search can often solve much larger problems than backtrack-based constraint systems. This has led to a variety of hybrids, and we briefly survey some of these.

A hybrid of the GSAT local search algorithm and the Dynamic Backtracking algorithm (Ginsberg & McAllester 1994) increases the flexibility in choice of backtracking variable. Local Changes (Verfaillie & Schiex 1994) is a complete backtracking algorithm that uses conflict analysis to unassign variables leading to constraint violation, and a heuristic similar to VH that restores assignments after backtracking. The timetabling algorithm of (Schaerf 1997) searches the space of all partial assignments. The Path-Repair Algorithm (Jussien & Lhomme 1999) is a generalisation of this approach that includes learning, allowing complete versions to be devised. The two-phase algorithm of (Zhang & Zhang 1996) searches a space of partial assignments, alternating backtracking search with local search. It can be tuned to different problems by spending more time in either phase. (Backer *et al.* 1997) generate partial assignments to key variables by local search, then pass them to a constraint solver that checks consistency. Large Neighbourhood Search (Shaw 1998) performs local search and uses backtracking to test the legality of moves. (Crawford 1993) uses local search within a complete SAT solver to select the best branching variable.

We should also mention variations on backtrack search that improve scalability by techniques other than local search. Iterative Sampling (Langley 1992) restarts a constructive search every time a dead-end is reached. Weak Commitment Search (Yokoo 1994) builds consistent partial assignments, using the min-conflict heuristic to guide value selection; on reaching a dead-end it restarts and uses learning to avoid redundant search. Bounded Backtrack Search (Harvey 1995) is a hybrid of Iterative Sampling and chronological backtracking, alternating a limited amount of chronological backtracking with random restarts. (Gomes, Selman, & Kautz 1998) periodically restart chronological or intelligent backtracking with slightly randomised heuristics. Limited Discrepancy Search (Harvey 1995; Harvey & Ginsberg 1995) searches the neighbourhood of a consistent partial assignment, trying neighbours in increasing order of distance from the partial assignment.

In the hybrid approach taken in this paper, a backtrack-based constraint algorithm is modified so that it is non-systematic. This can be viewed as local search in a space of consistent partial variable assignments, and demonstrably scales like more standard local search approaches (Prestwich 2000). So far this approach has only been applied to relatively simple propagation: forward checking on binary constraints (Prestwich 2002a), unit propagation for SAT problems (Prestwich 2002c; 2000), and a generalisation of unit propagation to 0/1 integer programs (Prestwich 2002b). However, its integration with more powerful propagation algorithms and other constraints is being investigated. Other researchers have applied randomised versions of backtracking, but unlike our approach have also aimed to preserve completeness (Lhomme 2002; Lynce & Marques-Silva 2002). Improved scalability is reported for these algorithms, but it is unclear whether local search-like scalability can be combined with completeness (except by using exponential memory).

Relaxation and local search

As in constraint programming, the surprisingly good scalability of local search has been noted in operations research. It has been successfully applied to many OR problems, but hybrids seem to be relatively rare. (Pesant & Gendreau 1996) use branch-and-bound to efficiently explore local search neighbourhoods. (Meyer 2000) integrates branch-and-bound into local search. (Meyer 1998) presents an approach that relies on repair-based search and a generic method for an exhaustive enumeration of repair steps, for solving partial constraint satisfaction problems exhaustively; this combines advantages of local search and extended branch-and-bound algorithms.

Non-systematic backtracking has also been applied to branch-and-bound. The problem of finding a low-autocorrelation binary sequence (LABS) is discussed in (Prestwich 2000). Previously, the only algorithm able to find optimal LABS was a branch-and-bound algorithm using a relaxation to compute lower bounds for the cost function. Several local search algorithms found large near-optimal solutions but failed to find even small optimal solutions. The hybrid performed local search in the state space of the branch-and-bound algorithm, found optimal solutions significantly faster, and found the best-known sequence of a certain size. Though this is the only such algorithm so far, we believe that the same technique will work with linear relaxations, yielding a local search algorithm for MIP that uses Simplex to prune its search space. However, in the next section we go further by adding constraint propagation.

Propagation, relaxation and local search

We now describe how to combine the three hybrid approaches discussed above into a new hybrid for MIP problems.

SAT

The starting point is the standard DLL (Davis-Logemann-Loveland) backtracking algorithm for SAT (propositional satisfiability). DLL interleaves backtracking with an inference rule called unit propagation, which can be viewed as constraint propagation. (Some versions of DLL add other inference rules.) Many modern systematic SAT solvers are versions of DLL, a recent and very fast example being Chaff (Moskewicz *et al.* 2001). Backtrack-based search sometimes scales poorly to large problems, but non-systematic search is a useful alternative. Though incomplete it often finds solutions far more quickly. A well-known local search algorithm for SAT is WSAT.

However, local search algorithms typically do not use unit propagation, making them unsuitable for some highly structured problems. This has motivated hybrid algorithms, including CLS (Prestwich 2002c; 2000) which is a non-systematic backtracking algorithm implemented for SAT as well as other problems. CLS scales like WSAT on a common benchmark (satisfiable random 3-SAT from the phase transition region) and performs well on highly structured problems, though WSAT is superior on other problems (Prestwich 2000).

ILP

SAT problems are known to be difficult for solution by branch-and-bound, though their encoding as 0/1 integer programs is trivial, motivating the adaptation of the SAT approach to OR problems. DLL has been generalised to ILP, with good results on a variety of problems (Barth 1995). Because of the success of local search on SAT problems, the WSAT local search SAT algorithm has also been generalised to ILP, again with good results (Walser 1997). CLS was recently generalised to ILP (Prestwich 2002b) and performed well on several problems. At the time of writing it has found the best-known solutions for several large instances of the Social Golfer problem; it matches the performance of the best-known algorithm for generating balanced incomplete block designs; it ranks approximately 6th of 33 algorithms tested on SAT-encoded hardware verification benchmarks; and it finds round-robin schedules beyond the reach of branch-and-bound (though it is not competitive with other algorithms on this problem).

The CLS implementation for ILP is called Saturn. It is described in (Prestwich 2002b) but here we give some details necessary for describing the new hybrid. Saturn begins like a standard backtracker by selecting a variable using a heuristic, assigning a value to it, performing constraint propagation where possible, then selecting another variable; on reaching a dead-end (a variable that cannot be assigned any value without causing domain wipe-out) it backtracks then resumes variable selection. Constraint propagation occurs in a linear constraint $\sum_{i=1}^{n} w_i l_i \leq d$ as follows. Any currently unassigned variable in the constraint whose weight is greater than $d - s$, where s is the sum of the weights of currently assigned literals, can have a domain value removed: 0 (false) if the literal is positive and 1 (true) if it is negative. This is a simple generalisation of the SAT propagation rule, which occurs when every literal but one has an assigned variable and is false, and where all weights are 1. A simple implementation trick speeds up execution: in a preprocessing phase before search begins, sort the variables in each constraint into order of decreasing weight. Then as soon as we find a variable in a constraint whose weight is no greater than $d - s$ we can ignore the rest of the variables in that constraint.

The novel feature of Saturn is the choice of backtracking variable: the variables selected for unassignment are chosen randomly, or using some other heuristic that does not preserve completeness. As in Dynamic Backtracking (Ginsberg 1993) only the selected variables are unassigned, without undoing later assignments. Because of this resemblance we call this form of backtracking *incomplete dynamic backtracking*. Unlike Dynamic Backtracking this algorithm is not complete, and we must sometimes force the unassignment of more than one variable. We do this by adding an integer *noise parameter* $B \geq 1$ to the algorithm and unassigning B variables at each dead-end. The only tricky part of the implementation concerns the combination of incomplete dynamic backtracking and constraint propagation. We omit these details, which are documented elsewhere.

MIP

For our MIP hybrid we add the Revised Simplex algorithm to Saturn. We anticipate no implementation difficulties: the hybrid will use an efficient, non-incremental version so that the linear relaxation can be computed at any point during search. The inputs are the constraints and the current set of variable assignments, and the output is either a flag stating that the current assignments are inconsistent, or a set of real-valued assignments to the unassigned variables.

It remains to decide when to apply Simplex. Care must be taken to avoid reduce overheads: in two recent hybrids of Simplex with constraint systems, Simplex was found to be expensive compared to the cost of constraint propagation. Best results were obtained in (Gomes & Shmoys 2002) by applying it only at selected points in the search tree, and in (Bosch & Trick 2002) by applying it only to the most important constraints. The latter technique is problem-dependent: for a given problem it must be decided which are the most relevant constraints. Instead we adapt the former technique to the CLS search algorithm as follows.

At the start of the search, Simplex is first computed to obtain a real value for each variable; these are used probabilistically to select 0/1 values for assignment. This randomised rounding approach (described in (Gomes & Shmoys 2002)) replaces the usual CLS value ordering heuristic, which simply aims to restore most previous assignments where possible. Variables are then selected and values assigned to them (guided by randomised rounding), applying constraint propagation after each assignment. On reaching a dead-end B selected variables are unassigned. So far the algorithm is simply Saturn, apart from the use of randomised rounding.

However, after unassigning B variables we reapply Simplex: if it detects inconsistency then a further B unassignments are made and Simplex is reapplied, and so on until no inconsistency is detected; then the algorithm starts to assign values to variables again, guided by the newly-computed real values.

Note that although the algorithm may temporarily move into regions that are LP-inconsistent, it immediately backs out of them; and the use of randomised rounding improves its chances of avoiding such regions. Note also that as well as controlling noise, B also controls how frequently Simplex is called: the larger the value of B the fewer calls will be made before LP-consistency is re-established. This is far cheaper than interleaving Simplex with variable assignment, and is analogous to the technique of only applying it up to a certain depth in the search tree. If the value of B is well chosen then Simplex may be called not much more than once per dead-end.

Discussion

Because non-systematic backtracking algorithms can scale like local search (Prestwich 2000) we claim that our new hybrid is in fact a local search algorithm. Its advantage over most local search approaches is that constraint propagation and Simplex are used to prune the search space, yielding a very tight integration of relaxation, propagation and local search.

We will initially apply the new hybrid to two problems: a pure ILP problem to evaluate the effects of Simplex on the existing algorithm, and a MIP problem to compare the new hybrid with other MIP approaches. The new hybrid is not yet implemented so this paper describes what we hope to achieve. We will present experimental results at the workshop.

References

Backer, B. D.; Furnon, V.; Kilby, P.; Prosser, P.; and Shaw, P. 1997. Search in constraint programming: Application to the vehicle routing problem. In *Constraint Programming 97, Proceedings of Workshop on Industrial Constraint-Directed Scheduling*.

Barth, P. 1995. A davis-putnam based enumeration algorithm for linear pseudo-boolean optimization. Technical Report mpi-i-95-2-003, Max-Plank Institut fur Informatik, Saarbrucken.

Bokmayr, A., and Kasper, T. 1998. Branch and infer: A unifying framework for integer and finite domain constraint programming. *INFORMS Journal on Computing* 10.

Bosch, R., and Trick, M. 2002. Constraint programming and hybrid formulations for three life designs. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems*, 77–92.

Brockmann, K., and Decker, T. 2000. A parallel tabu search algorithm for short term lot sizing and scheduling in flexible flow line environments. In *Sixteenth International Conference on CAD/CAM, Robotics and Factories of the Future*.

Caseau, Y., and Laburthe, F. 1995. Improving branch and bound for jobshop scheduling with constraint propagation. *Combinatorics and Computer Science* 129–149.

Crawford, J. M. 1993. Solving satisfiability problems using a combination of systematic and local search. In *Second DIMACS Challenge: Cliques, Coloring, and Satisfiability*.

Do, M. B., and Kambhampati, S. 2000. On the use of lp relaxations in solving sat encodings of planning problems. In *AAAI 2000 AI/OR workshop*.

Focacci, F.; Lodi, A.; and Milano, M. 1999. Cost-based domain filtering. In *International Conference on Principles and Practice of Constraint Programming*, 189–203. Springer-Verlag.

Ginsberg, M. L., and McAllester, D. A. 1994. Gsat and dynamic backtracking. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, 226–237. Springer-Verlag.

Ginsberg, M. L. 1993. Dynamic backtracking. *Journal of Artificial Intelligence Research* 1:25–46.

Gomes, C. P., and Shmoys, D. 2002. The promise of lp to boost csp techniques for combinatorial problems. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems*, 291–306.

Gomes, C.; Selman, B.; and Kautz, H. 1998. Boosting combinatorial search through randomization. In *Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, 431–437. AAAI Press/The MIT Press.

Gomes, C. 2000. Structure, duality, and randomization: Common themes in ai and or. In *Seventeenthth National Conference on Artificial Intelligence*.

Hajian, T.; El-Sakkout, H.; Wallace, M.; Lever, J.; and Richards, E. 1998. Towards a closer integration of finite domain propagation and simplex-based algorithms. *Annals of Operations Research*.

Harvey, W. D., and Ginsberg, M. L. 1995. Limited discrepancy search. In *Fourteenth International Joint Conference on Artificial Intelligence*, 607–615.

Harvey, W. D. 1995. *Nonsystematic Backtracking Search*. Ph.D. Dissertation, Stanford University.

Hooker, J. N. 2000. Logic, optimization, and constraint programming. *INFORM Journal of Computing*.

Jussien, N., and Lhomme, O. 1999. The path-repair algorithm. In *Workshop on Large Scale Combinatorial Optimization and Constraints*, volume 4 of *Electronic Notes in Discrete Mathematics*.

Langley, P. 1992. Systematic and nonsystematic search strategies. In *First International Conference on Artificial Intelligence Planning Systems*.

Lhomme, O. 2002. Amortized random backtracking. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CPAIOR-02)*, 21–32.

Lynce, I., and Marques-Silva, J. P. 2002. Complete unrestricted backtracking algorithms for satisfiability. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing*.

Meyer, H. 1998. Finding regions for local repair in partial constraint satisfaction. In *Advances in Artificial Intelligence, 22nd Annual German Conference on Artificial Intelligence*, 57–68. Springer Verlag.

Meyer, H. 2000. Solving rostering tasks as constraint optimization. In Burke, and Erben., eds., *Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling (PATAT-2000)*, Lecture Notes on Computer Science. Springer Verlag.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *Thirty-Ninth Design Automation Conference, Las Vegas*.

Nemhauser, G. L., and Wolsey, L. A. 1988. *Integer and Combinatorial Optimization*. New York: John Wiley & Sons, Inc.

Pesant, G., and Gendreau, M. 1996. A view of local search in constraint programming. In *Second International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*, 353–366. Springer Verlag.

Prestwich, S. D. 2000. A hybrid search architecture applied to hard random 3-sat and low-autocorrelation binary sequences. In *Sixth International Conference on Principles and Practice of Constraint Programming*, volume 1894 of *Lecture Notes in Computer Science*, 337–352. Springer-Verlag.

Prestwich, S. D. 2002a. Coloration neighbourhood search with forward checking. *Annals of Mathematics and Artificial Intelligence* 34:327–340.

Prestwich, S. D. 2002b. Randomised backtracking for linear pseudo-boolean constraint problems. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CPAIOR-02)*, 7–20.

Prestwich, S. D. 2002c. Sat problems with chains of dependent variables. *Discrete Applied Mathematics*. to appear.

Rodosek, R.; Wallace, M. G.; and Hajian, M. T. 1999. A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operational Research, Recent Advances in Combinatorial Optimization* 86:63–87.

Schaerf, A. 1997. Combining local search and look-ahead for scheduling and constraint satisfaction problems. In *Fifteenth International Joint Conference on Artificial Intelligence*, 1254–1259.

Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming, Proceedings of the Fourth International Conference*, volume 1520 of *Lecture Notes in Computer Science*, 417–431. Springer-Verlag.

Thorsteinsson, E. S. 2001. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *Seventh International Conference on Principles and Practice of Constraint Programming (CP2001), Paphos, Cyprus*.

Verfaillie, G., and Schiex, T. 1994. Solution reuse in dynamic constraint satisfaction problems. In *Twelfth National Conference on Artificial Intelligence, AAAI Press*, 307–312.

Walser, J. P. 1997. Solving linear pseudo-boolean constraints with local search. In *Eleventh Conference on Artificial Intelligence (AAAI'1997)*, 269–274.

Wolfman, S. A., and Weld, D. S. 1999. Combining linear programming and satisfiability solving for resource planning. *Knowledge Engineering Review, special issue on Artificial Intelligence and Operations Research* 15.

Yokoo, M. 1994. Weak-commitment search for solving constraint satisfaction problems. In *Twelfth National Conference on Artificial Intelligence*, 313–318. AAAI Press.

Zhang, J., and Zhang, H. 1996. Combining local search and backtracking techniques for constraint satisfaction. In *Thirteenth National Conference on Artificial Intelligence and Eighth Conference on Innovative Applications of Artificial Intelligence*, 369–374. AAAI Press/MIT Press.