Solving Linear Pseudo-Boolean Constraint Problems with Local Search

Joachim P. Walser
Programming Systems Lab
Universität des Saarlandes, Postfach 151150
66041 Saarbrücken, Germany
walser@ps.uni-sb.de

Abstract

Stochastic local search is one of the most successful methods for model finding in propositional satisfiability. However, many combinatorial problems have no concise propositional encoding. In this paper, we show that domain-independent local search for satisfiability (Walksat) can be generalized to handle systems of linear pseudo-Boolean (0-1 integer) constraints, a representation that is widely used in operations research. We introduce the algorithm WSAT ($\mathcal{PB}$) and demonstrate its potential in two case studies. The first application is an optimization problem from radar surveillance. Experiments on problems of realistic size show that WSAT ($\mathcal{PB}$) is an efficient heuristic to find good approximate solutions. For most of the test problems, it found provably optimal solutions. In the second case study, we show that pseudo-Boolean local search can efficiently solve the progressive party problem, a problem that is hard for constraint programming with chronological backtracking, and whose 0-1 encoding is beyond the size limitations of integer linear programming.

Introduction

Local search is versatile and many successful applications of domain-specific local search methods have been reported. Further, a number of successful domain-independent methods exist that include strategies for maximum satisfiability (Hansen & Jaumard 1990), for certain realistic constraint satisfaction (CSP) problems (Minton *et al.* 1990; Hao & Dorne 1996), and some of the most efficient methods for hard realistic and randomly generated propositional satisfiability (SAT) problems (Selman, Levesque, & Mitchell 1992; Selman, Kautz, & Cohen 1994; Selman & Kautz 1996).

However, many combinatorial problems have no concise propositional encoding and hence local search algorithms for SAT cannot be applied. Nonetheless, many of these problems can be modeled concisely with linear 0-1 integer inequalities (pseudo-Boolean constraints). Pseudo-Boolean constraints have a long tradition in operations research (OR) and cover many problems like scheduling, sequencing, time-tabling, etc. While in OR these problems are typically solved with methods based on linear programming (Nemhauser & Wolsey 1988), we show in this paper that stochastic local search can be an alternative for discrete 0-1 feasibility problems and, using over-constrained systems, for certain optimization problems as well.

We generalize stochastic local search for satisfiability, Walksat (Selman, Kautz, & Cohen 1994), to handle linear pseudo-Boolean constraint systems and introduce the algorithm WSAT ($\mathcal{PB}$). Only a small decrease in the number of local moves (variable flips) per second is incurred by this generalization. WSAT ($\mathcal{PB}$) inherits the variable selection strategy of Walksat and is equipped with a history mechanism (Gent & Walsh 1993) and a short tabu memory (see (Glover & Laguna 1993) for an overview of tabu search).

To explore the potential of local search for this constraint class we perform two case studies on realistic problems. We first consider an optimization problem from the radar surveillance domain which we model as an over-constrained system using hard and soft constraints. For all test problems, WSAT ($\mathcal{PB}$) efficiently finds good approximate solutions whereas problems of realistic size appear to be beyond exact optimization with integer programming branch-and-bound. Linear programming (LP) relaxations can provide lower bounds on the solution quality and for many of the test problems optimality of the local search solutions can be proved.

The problem in the second case study, "progressive party," was recently introduced in a comparison between constraint programming and integer linear programming (Smith *et al.* 1996). A main result of the study is that the problem appears to be beyond the size limitations of integer linear programming (ILP) but can be solved using constraint propagation and chronological backtracking. Our experiments show that the problem can be solved significantly faster using WSAT ($\mathcal{PB}$). Further, we looked at slight random variations of the instance given in (Smith *et al.* 1996) and found that local search was robust with respect to the modifications. On the other hand we were not able to find a constraint program that solved all test problems. To solve the problem with WSAT ($\mathcal{PB}$), we factor it into two stages. In the first stage, a small number of principal variables are explicitly enumerated while in the second stage the remaining subproblem is attacked with local search.

Pseudo-Boolean Constraints

Many combinatorial problems have no concise encoding in propositional logic but can be stated concisely using slightly richer constraints. For example, consider a statement of the pigeonhole problem with Boolean variables p_{ij}, where p_{ij} means pigeon i is in hole j. A natural encoding is to use two different constraints, (a) every pigeon is in exactly one hole $\sum_j p_{ij} = 1$ (for all i), and (b) no two pigeons are in the same hole $\sum_i p_{ij} \leq 1$ (for all j). Given n pigeons and m holes, this formulation consists of $n + m$ pseudo-Boolean clauses. On the other hand, a SAT encoding would be (a) $\vee_j\ p_{ij}$ (for all i) and $\forall j \forall k \neq j : p_{ij} \rightarrow \overline{p_{ik}}$ (for all i); similarly for (b). With $O(m^2 n + n^2 m)$ clauses, the size of this SAT encoding would be impractical for larger instances.

The class of linear pseudo-Boolean constraints is defined as follows (Hammer & Rudeanu 1968). A *linear pseudo-Boolean clause* has the form

$$\sum_{i \in I} c_i \cdot L_i \sim d,$$

where $c_i, d \in \mathbb{Z}$, $\sim \in \{=, \leq, <, \geq, >\}$, and the L_i are literals for all $i \in I$ (a literal is a Boolean variable or its negation). Pseudo-Boolean constraints generalize SAT in the sense that every Boolean clause (disjunction of literals) can be translated into a single linear pseudo-Boolean inequality but not vice versa.

In general, converting linear pseudo-Boolean constraints to propositional satisfiability can largely increase the size of the encoding. Additionally, algorithms like Walksat require input formulas to be in conjunctive normal form (CNF), which may cause a further (exponential) increase in the size of the formula or an enlargement of the search space through additional variables. Alternatively, CNF conversion can be avoided at the cost of a decreased flip-rate (Sebastiani 1994). In the following section, we will extend propositional local search (Walksat) to handle pseudo-Boolean constraint problems.

The WSAT $(\mathcal{PB})$ procedure

WSAT $(\mathcal{PB})$ performs randomized greedy local search on linear pseudo-Boolean constraints, equipped with a history mechanism and a tabu memory. Figure 1 gives the outline of a general local search routine (Selman, Levesque, & Mitchell 1992) to find a satisfying assignment for a set of constraints α. Local moves of the search are the "flips" of Boolean variables that are chosen by *select–variable* according to a randomized greedy strategy. The parameter *Max-Flips* governs restarts while *Max-Tries* ensures termination. *New* assignments are chosen by a biased random function such that with probability p_z, a variable is assigned 0 and with probability $1 - p_z$ it is assigned 1. This can help reducing the number of initially violated clauses but did not appear to be critical in our experiments.

```
proc local-search
  input clauses α, Max-Flips, Max-Tries
  for i := 1 to Max-Tries do
      A := new total truth assignment
      for j := 1 to Max-Flips do
          if A satisfies α then return A
          P := select–variable(α, A)
          A := A with P flipped
      end
  end
  return "No satisfying assignment found"
end
```

Figure 1: A generic local search procedure for SAT or linear pseudo-Boolean clauses.

Variable selection. This section describes a carefully engineered variable selection strategy for WSAT $(\mathcal{PB})$ that originates from Walksat's variable selection (Selman & Kautz 1996). It is capable of handling systems of linear pseudo-Boolean constraints with hard and soft constraints.

First, an unsatisfied clause is selected as follows. If at any time only hard or only soft unsatisfied clauses exist, an unsatisfied clause is picked at random. If both hard and soft unsatisfied clauses exist, with probability p_{hard} a random unsatisfied hard clause and with $1 - p_{hard}$ a soft clause is selected. Next, we need to decide which variable to flip from the clause. Notice that unlike for propositional Walksat, the selected clause may remain unsatisfied when flipping a variable in it. Therefore, with every flip, we attempt to reduce the overall *score* which we define as follows: For an assignment A, let $d(c, A)$ be the *net integer distance* of clause c from being satisfied. If c is satisfied under A, $d(c, A) = 0$. The *score* of an assignment A is the sum of the scores of all clauses. If an assignment is satisfying, its score is 0. Empirically, a good rule appears to be to choose the variable that decreases the score most, if there is one that decreases it. Otherwise, with a given probability p_{noise}, choose a random variable from the clause, and with $1 - p_{noise}$, choose the variable that increases the score least.

Additionally, WSAT $(\mathcal{PB})$ is equipped with a tabu memory (Glover & Laguna 1993) of size t: No proposition may be flipped that has been flipped in the previous t moves. Further, all ties between variables are broken by a history mechanism (Gent & Walsh 1993): On ties, the history mechanism chooses the variable that was flipped longest ago. In the greedy branch, there is a tie between two variables if flipping either one causes equal score improvement. In the noise branch, the history mechanism avoids randomness and uses non-random diversification instead since there is a tie between all variables in the selected clause.

Unlike for many propositional problems, the algorithm appeared to perform best if run in a very greedy way (we set $p_{noise} = 0.01$). Extending Walksat to handle clauses with different relational operators, coefficients, and addition instead of disjunction incurs only a relatively small overhead over the propositional case. In the following, we present two case studies that apply WSAT $(\mathcal{PB})$ to realistic problems.[1]

[1] The encodings of all sample instances are available through http://www.cirl.uoregon.edu/constraints/.

Case study I: Radar surveillance

The following problem and its modeling originate from a project currently carried out at the Swedish Institute of Computer Science (SICS) (Brand, Haridi, & Olsson 1997). The problem is to allocate a number of radar stations for observation of a geographic area, such that each point is observed by at least three stations. As customary in the radar surveillance domain, the area is divided into hexagonal cells and radar stations are located in a number of cells. Each radar station can divide its signal scope circle into six sectors and each station can vary the signal strength in each sector independently from zero to some given maximum distance d_{max}. Each cell must be covered by 3 radar stations (desired coverage *dc*=3) and all coverage beyond this is to be minimized (*over-coverage*) for economic reasons and for detectability.[2] Figure 2 gives an illustration.

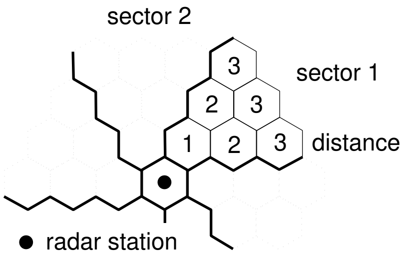

Figure 2: Radar map with hexagonal cells.

The problem can be encoded by the following over-constrained pseudo-Boolean system. For every combination of radar station u, sector $1 \leq s \leq 6$ and possible observation distance $1 \leq d \leq d_{max}$, a Boolean variable σ_{usd} is introduced. Variable $\sigma_{usd} = 1$ if and only if station u is switched on in sector s at distance d. The set of all cells that station u can reach in sector s at distance d is denoted by C_{usd}. The constraints are the following.

Constraints MC. There are significant and insignificant cells. Significant cells must be covered by at least three stations

$$\sum_{c \in C_{usd}} \sigma_{usd} \geq 3 \text{ for all } c$$

Similarly, insignificant cells have desired coverage *dc*=0.

Consistency constraints PC. If station u is switched on at distance $d > 1$ in sector s, it must also be switched on at distance $d-1$ in sector s.

$$\sigma_{usd} - \sigma_{usd'} \leq 0 \text{ for all } u, s, 1 < d \leq d_{max};\ d' = d-1$$

Soft constraints OC. Cells should not be covered by more than three stations.

$$\sum_{c \in C_{usd}} \sigma_{usd} \leq 3 \text{ for all } c$$

[2] Cells that physically cannot be covered by at least three stations must be covered by as many stations as possible and are then factored out. In the original model a radar station can be switched on to cover only the cell it is located in and always covers it provided it is switched on for some sector.

Minimizing over-coverage. In order to minimize the total over-coverage, minimizing the number of violated soft constraints is not sufficient because over-coverage can occur to different degrees for each cell. However, because there is one soft clause for each cell, the over-coverage of the system corresponds exactly to the score that is minimized by WSAT $(\mathcal{PB})$.

To solve the problem with linear/integer programming, the soft constraints were replaced by an objective function that measures total over-coverage (n_s is the number of significant cells),

$$\min \left(\sum_{c} \sum_{c \in C_{usd}} \sigma_{usd}\right) - 3\, n_s$$

Experimental results

Table 1 summarizes the experimental results. Based on a sample of radar instances (generated at SICS), we ran WSAT $(\mathcal{PB})$ and a commercial integer branch-and-bound algorithm (CPLEX). All instances were randomly generated and vary in size (100 to 2100 cells), in the percentage of insignificant cells (0% and 2%), and in the spread of radar stations (even or uneven). The density of stations remained constant.

Of course, it is in general not possible to compare a heuristic with a complete algorithm because the latter is guaranteed to find optimal solutions while the former is not. However, many instances of the sample have the characteristic that the optimal value of the over-coverage is the same as the LP lower bound.[3] These lower bounds can be efficiently computed with linear programming and provide a guaranteed bound for the quality of solutions found by WSAT $(\mathcal{PB})$. According to Haridi *et al.*, the long-term goal of the project is to cover a large geographical area with thousands of cells. It is thus an important criterion of success that the solution strategy scales well. From the experiments we see that for smaller problems branch-and-bound performs well, while it appears that solving realistic instances optimally will be infeasible. Scaling of local search has previously been examined on hard randomly generated satisfiability problems where sub-exponential (average) scaling was observed (Gent & Walsh 1993; Parkes & Walser 1996).

At any point during the search the lower bounds can give a provable maximum deviation from the optimal over-coverage. To improve the runtimes of WSAT $(\mathcal{PB})$, it could be cut off as soon as the lower bound is reached. Conversely, branch-and-bound could be cut off at any point with a near-optimal solution and a comparison could be done based on fixed allowed runtimes. For WSAT $(\mathcal{PB})$, a history mechanism appeared to be critical for success; a tabu memory of size 1 was used and variables were initialized with $p_z = 0.5$. Hard clauses were always prefered for repair ($p_{hard} = 1.0$). CPLEX was run with standard parameter settings which is arguably unfair but partly compensated by the fact that CPLEX is the product of years of development.

[3] We thank Alexander Bockmayr for pointing this out.

size	n	m	spread	$\%$sig	LP lb	WSAT ($\mathcal{PB}$)			CPLEX	
						best	fast	time/s	oc*	time/s
100:22	434	606	even	100	0	0	0.0	0.1	0	2
200:44	933	1273	even	100	1	1	1.0	0.6	1	23
900:200	4616	6203	even	100	2	2	2.0	18.3	2	3356
2100:467	10975	14644	even	100	3	3	3.5	41.2	3	–
100:22	410	581	even	98	1	1	1.1	1.4	1	2
200:44	905	1246	even	98	2	2	2.0	4.0	2	17
900:200	4623	6174	even	98	4	6	8.0	25.9	4	1653
2100:467	10989	14595	even	98	11.5	19	23.2	62.0		–
100:22	371	518	uneven	100	3	3	3.0	1.1	3	2
200:44	772	1065	uneven	100	0	0	0.1	0.6	0	16
900:200	4446	5699	uneven	100	5	5	5.0	18.8	5	4134
2100:467	10771	14002	uneven	100	8.1	14	16.6	43.2		–

Table 1: Experimental comparison on radar surveillance problems: Columns are problem size in number of cells and stations (stations have a maximal reach of $d_{max} = 4$), encoding size in number of variables n and clauses m, spread of stations on the map, percentage of significant cells, and LP lower bound for over-coverage. 'Fast' measures best over-coverage found in 30K, 100K, 300K, and 500K flips respectively (averaged over 20 runs), 'time' gives average time needed. 'Best' measures best over-coverage found within all 20 runs. CPLEX columns are optimal over-coverage and runtime. '–' means not optimally solved in hours of computation. All runtimes measured on a SPARCstation 20.

Experiments with constraint programming. Various models (finite domain integer and Boolean) and enumeration schemes have been tried (Brand, Haridi, & Olsson 1997). Although small problems are solved to optimality quickly, the larger sample instances can be solved within reasonable time with large over-coverage only. We hypothesize that it is thrashing that makes these problems hard for a constraint program that backtracks chronologically: Two distant radar stations hardly affect each other, yet with chronological backtracking the state of one station is only changed after visiting the complete subspace of configurations of many other stations.

One could argue that the success of local search on the radar domain is caused by the geometric locality of the influence of each radar station. The motivation for the second case study is to look at a hard constraint problem that does not have this property.

Case study II: Progressive party

Recently, a comparison study between integer linear programming and constraint programming has been performed based on a problem called the "progressive party problem" (Smith *et al.* 1996). The 0-1 modeling of this problem is large and uses a variety of different constraints which suggested that it would be an interesting test case for pseudo-Boolean local search and could indicate whether the results from radar surveillance could carry over to other domains. The problem scenario is an evening party in the context of a yachting rally. Certain boats are selected to be hosts, and the crews of the remaining boats in turn visit the host boats for several successive half-hour periods. The crew of a host boat remains on board to act as hosts while the crew of a guest boat together visits several hosts. Every boat can only host a limited number of guests at a time and crew sizes are different. The table with boat capacities and crew sizes can be found in (Smith *et al.* 1996); there are six time periods. A guest boat cannot revisit a host and guest crews cannot meet more than once. The problem facing the rally organizer is that of minimizing the number of host boats: Certain boats are constrained to be hosts, and selecting the hosts among the remaining boats is stated as part of the problem.

We do not claim that this problem is of immediate practical significance; however, it has the advantage of being a well-studied hard assignment problem with a variety of constraints. The variables in the problem are the following: $\delta_i = 1$ iff boat i is used as host boat. Variables $\gamma_{ikt} = 1$ iff boat k is a guest of boat i in period t. Constant c_i is the crew size of boat i and K_i is its total capacity. The objective is to minimize the number of hosts $\sum_i \delta_i$, subject to:

Constraints CD. A boat can only be visited if it is a host boat.

$$\gamma_{ikt} - \delta_i \leq 0 \text{ for all } i, k, t;\ i \neq k.$$

Constraints CCAP. The capacity of a host boat cannot be exceeded.

$$\sum_{k, k \neq i} c_k \gamma_{ikt} \leq K_i - c_i \text{ for all } i, t.$$

Constraints GA. Each crew must always have a host or be a host.

$$\delta_k + \sum_{i, i \neq k} \gamma_{ikt} = 1 \text{ for all } k, t.$$

Constraints GB. A guest crew cannot visit a host boat more than once.

$$\sum_t \gamma_{ikt} \leq 1 \text{ for all } i, k;\ i \neq k.$$

An additional set of 0-1 variables was introduced to state the meet-once restrictions. $m_{klt} = 1$ if boats k and l meet at time t. This simplifies the ILP model described in (Smith *et al.* 1996).[4]

Constraints U. Link m_{klt} with γ_{ikt}. $\gamma_{ikt}+\gamma_{ilt}-m_{klt} \leq 1$ for all $k, l, t;\ k < l$.

Constraints M. Every pair of hosts can meet at most once.

$$\sum_t m_{klt} \leq 1 \text{ for all } k, l;\ k < l.$$

With B boats and T time periods, the problem has $O(B^2T)$ variables and $O(B^2T)$ constraints in this formulation. Smith *et al.* note that the CP representation is more compact and has "far fewer constraints and variables than the ILP". This is not the case since the number of both constraints and variables is actually $O(B^2T)$ in both encodings (even in the improved ILP model in (Smith *et al.* 1996)).

Although the problem is formulated as an optimization problem, given the particular description of the participating boats the task is to find a feasible assignment with 13 host boats. Every solution with 13 hosts is optimal because the capacity constraints cannot be met with 12 hosts even for a single time period. Solving the problem can be divided into two stages: (i) selection of the host boats, and (ii) assignment of guest boats to hosts for all time periods. It turns out that the *spare capacity* of the boats is a good indicator of whether a boat should be host or guest, so after forcing special boats to be hosts (e.g. the rally organizer), the remaining hosts were selected by decreasing spare capacity (the spare capacity of a boat is its total capacity minus its crew size). In both the ILP and the CP approach, Smith *et al.* treat both stages of the problem. However, the search-space for a particular host selection is too large to be explored exhaustively within hours of computation. This shows that solving stage (ii) by itself is a hard subproblem and we therefore focus on stage (ii): Finding a guest allocation given a fixed selection of hosts. Thereafter we will outline a strategy that captures both stages.

Smith *et al.* report the problem could not be solved with a commercial integer programming tool (XPRESSMP, using a variety of tricks) because it appears to be beyond the size limitations of ILP.

Experimental results

For the experiments, we used the original problem instance of Smith *et al.* and randomly varied the host selection to produce 5 additional instances. For all instances, we kept the original description of boat capacities and crew sizes. After fixing the 13 hosts and performing constraint propagation as an efficient preprocessing, the original problem has 4632 variables and 30965 remaining clauses in pseudo-Boolean formulation. WSAT ($\mathcal{PB}$) finds a feasible guest allocation in 5.5 seconds (averaged over 20 successful runs on a SPARCstation 20) using a tabu memory of size 1 and initializing with a bias of $p_z = 0.9$. Additionally, setting up the constraints from an abstract representation requires around 15 seconds. Table 2 summarizes the results.

host boats	h	g	%cap	WSAT ($\mathcal{PB}$)
1–12,16	100	92	.92	2.9s
1–13 (orig)	98	94	.96	5.5s
1,3-13,19	96	92	.96	6.4s
3–13,25,26	98	94	.96	8.8s
1–11,19,21	95	93	.98	31.6s
1–9,16–19	93	91	.98	42.5s

Table 2: Empirical comparison on variations of the progressive party problem. The columns are: Selected hosts, total sum of host spare capacities h, total sum of guest crew sizes g; percentage of total capacity used as a measure of constrainedness (%cap $= g/h$). Runtimes averaged over 20 runs of WSAT ($\mathcal{PB}$), *Max-Flips*$=\infty$, flip-rate 1.1 K-flips/s.

For comparison, Smith *et al.* report 27 minutes of runtime of their ILOG Solver program on a SPARCstation IPX. To reproduce the results, we implemented the described modeling in Oz, a concurrent constraint language (Smolka 1995).[5] We used constraints and a labelling strategy similar to the one described by Smith *et al.*. Although our constraint program was able to solve the original instance in 8 minutes (on a SPARCstation 20), we could not find a labelling strategy that was able to solve all sample instances.[6]

Embedding into constraint programming. To solve both stages of the problem, we propose a loose coupling of systematic and local search. The approach simply enumerates the principal variables heuristically (in this case the δ_i's, stage (i)), then performs constraint propagation/simplification and applies local search to solve the remaining subproblem (stage (ii)). In our implementation, we use an embedding of WSAT ($\mathcal{PB}$) into the constraint language Oz. The advantage of using a constraint language is the high-level support for problem modeling and solution checking. Oz additionally offers the use of computation spaces which simplifies the embedding of a solver like WSAT ($\mathcal{PB}$) into CP.

Historic remarks. Before using the two-stage approach, we experimented with local search on a reduced version of the problem that included host selection. Observation of the local search process revealed that host selection and guest allocation were mixed and hosts were changed almost as often as the guest allocation was repaired. This seemed to be an unreasonable strategy.

Related Work

The tabu search meta heuristic has been applied to find good initial feasible solutions for 0-1 integer programs. However, in most examples the subroutine is domain specific. Based on the pivot and complement heuristic (Balas & Martin

[4]The original ILP description (Smith *et al.* 1996) is $m_{klt} = 1$ **iff** boats k and l meet at time t. The modification simplifies the problem and saves approximately 30K clauses. According to Sally Brailsford (personal communication) this had been tried in the ILP model.

[5]Publically available from http:://www.ps.uni-sb.de/oz/.

[6]Jörg Würtz, personal communication.

1980), a general 0-1 integer tabu search has been proposed (Aboudi & Jörnsten 1994). The problems reported were hard optimization problems with up to 90 variables and 30 constraints. Our approach is different in that WSAT $(\mathcal{PB})$ aims at finding solutions to hard, possibly over-constrained feasibility problems without an explicit objective function. Local moves of WSAT $(\mathcal{PB})$ are exclusively variable flips and tabu memory structures are simple. We would be interested in a comparison of the approaches.

Pseudo-Boolean constraints have been introduced into the framework of constraint logic programming with the language CLP$(\mathcal{PB})$ where they can be solved with logic-based methods (Bockmayr 1992).

Conclusions

We have generalized the Walksat algorithm to handle systems of linear pseudo-Boolean constraints and introduced the algorithm WSAT $(\mathcal{PB})$. In two case studies, we have applied this algorithm to 0-1 integer programming problems, radar surveillance and progressive party, and find its performance to be superior to existing techniques for these domains. This shows that domain-independent local search can successfully solve more expressive constraint systems than SAT and CSP. It opens the field to attack those pseudo-Boolean constraint problems with stochastic local search for which a compilation to propositional logic is problematic. For the two domains we considered, we found that little customization was necessary and that parameter settings were similar. What seems to matter beyond a carefully engineered variable selection was a high degree of greediness, a history mechanism, and a short tabu memory.

From the OR perspective, further investigation is needed to assess how WSAT $(\mathcal{PB})$ will perform as a primal heuristic for discrete optimization in general. We have shown that if the objective function can be expressed with soft constraints (case study I), or if the problem can be stated as a short sequence of feasibility problems (case study II), local search can outperform its competition.

Acknowledgments

The author is supported by a DFG (Deutsche Forschungsgemeinschaft) doctoral fellowship. Many thanks to Seif Haridi and Per Brand for providing the radar surveillance problems which sparked this work. The author is grateful to Martin Henz and Gert Smolka for illuminating discussions of this work and for comments on an earlier draft. Many thanks to Jörg Würtz and Thorsten Ölgart for modeling the progressive party problem in Oz, and to Alexander Bockmayr and Thomas Kasper for providing insights into optimization issues. Thanks also to Andrew Parkes and the anonymous reviewers for helpful suggestions.

References

Aboudi, R., and Jörnsten, K. 1994. Tabu search for general zero-one integer programs using the pivot and complement heuristic. *ORSA Journal on Computing* 6(1):82–93.

Balas, E., and Martin, C. 1980. Pivot and complement – a heuristic for zero-one programming. *Management Science* 26:86–96.

Bockmayr, A. 1992. Logic programming with pseudo-boolean constraints. In Colmerauer, A., and Benhamou, F., eds., *Constraint Logic Programming – Selected Research*. MIT Press.

Brand, P.; Haridi, S.; and Olsson, O. 1997. Some radar surveillance problems. Technical report, Swedish Institute of Computer Science, SICS. To appear.

Gent, I., and Walsh, T. 1993. Towards an understanding of hill-climbing procedures for SAT. In *Proceedings AAAI-93*, 28–33.

Glover, F., and Laguna, M. 1993. Tabu search. In Reeves, C. R., ed., *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press. chapter 3, 70–150.

Hammer, P., and Rudeanu, S. 1968. *Boolean Methods in Operations Research and Related Areas*. Springer.

Hansen, P., and Jaumard, B. 1990. Algorithms for the maximum satisfiability problem. *Computing* 44:279–303.

Hao, J.-K., and Dorne, R. 1996. Empirical studies of heuristic local search for constraint solving. In *Proceedings CP-96*, 194–208.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1990. Solving large-scale constraint satisfcation and scheduling problems using a heuristic repair method. *Artificial Intelligence* 58:161–205.

Nemhauser, G., and Wolsey, L. 1988. *Integer and Combinatorial Optimization*. Series in Discrete Mathematics and Optimization. Wiley-Intersience.

Parkes, A., and Walser, J. 1996. Tuning local search for satisfiability testing. In *Proceedings AAAI-96*, 356–362.

Sebastiani, R. 1994. Applying GSAT to non-clausal formulas. *JAIR-94* 1:309–314.

Selman, B., and Kautz, H. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings AAAI-96*, 1194–1201.

Selman, B.; Kautz, H.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proceedings AAAI-94*, 337–343.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings AAAI-92*, 440–446.

Smith, B.; Brailsford, S.; Hubbard, P.; and Williams, H. 1996. The progressive party problem: Integer linear programming and constraint programming compared. *Constraints* 1:119–138.

Smolka, G. 1995. The Oz programming model. In *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000. Berlin: Springer-Verlag. 324–343.