BnB vs Label Relaxation.

Project report

By

Dima Vingurt
Table of Contents

Aim of the project. ..........................................................................................................................3
CSP and WCSP problems...........................................................................................................3
BnB ..............................................................................................................................................3
Relaxation Labeling (RL) .........................................................................................................4
Implementation..............................................................................................................................4
  Problem ..................................................................................................................................4
  BnB ..........................................................................................................................................4
  RL............................................................................................................................................5
Experiments and results..............................................................................................................5
Conclusions .................................................................................................................................6
Reference .....................................................................................................................................7
**Aim of the project.**

Implement and test relaxation labeling (RL) algorithm on constrain optimization problems (COP). Branch and bound (BnB) algorithm used as reference for best/optimal solution.

**CSP and WCSP problems.**

*Constraint satisfaction problems* (CSP) are problems defined as a set of objects whose state must satisfy a number of constraints. CSP represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods. CSPs often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods to be solved in a reasonable time. The boolean satisfiability problem (SAT) can be CSP. "Real life" examples include planning and resource allocation.

CSP can be represented as set of \( \{V,D,C\} \). Where \( V \) is group of variables, \( D \) is domains for each variables and \( C \) is constrains between variables. The solution is assignment for each variable from its domain. The assignment must satisfy all constrains.

Weighted constraint satisfaction problem (WCSP): in addition to standard CSP the weight (or price) for each constrain (also possible to add price for some assignments) can be added. Solution will be the optimal (minimum or maximum of total price) assignment.

**BnB**

In order to solve CSP we can apply backtracking (BT) algorithm. BT is a general algorithm for finding all (or some) solutions to some computational problems, notably CSP, that incrementally builds candidates to the solutions, and abandons each partial candidate ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution. BT can be easily visualized as search on the problem tree as shown on Figure 1. Important to add, that variables have order, which doesn’t change. Order on figure 1 is variable X and then Y. If Y’s domain is empty – choose next value in X’s domain.

![Figure 1: Search Tree for CSP](image)

Let’s consider the WCSP problem with price for each constrain and then let’s find the optimal solution with minimal price. BT is not suited for this task. In order to solve constrain optimization problem (COP) we need to visit each node and compere the total price (sum of all “broken” constrains). BnB
algorithm is extended version of BT, which uses global value “ub” - upper bound. “Ub” initiated with infinity value. When leaf is reached in our search: compare total price with “ub”; if “ub” > total price then “ub” <- total price and memorize this assignment (this will be current optimal assignment). When visiting each node: compare total partial cost of current partial assignment with “ub”. If total partial cost exceeds “ub” then do backtracking.

BnB is complete algorithm- that’s mean that solution found by BnB is optimal.

Relaxation Labeling (RL)

Another way to solve COP problems is to try local search algorithms. RL is such algorithms. In order to solve COP with RL we need to define set of objects, labels, compatibility, support function and update rule.

Set of objects will be variables and labels will be values. It is logically to assume compatibility as constrains. \( r_{ij}(\lambda, \mu) \) - will be the price of constrain variable i with value \( \lambda \) and variable j with value \( \mu \). In order to define support we have to remember, that we are looking for local minimum of price:

\[
S_i(\lambda) = \sum_{j=1}^{n} \sum_{\mu=1}^{m} \left( 1 / r_{ij}(\lambda, \mu) \right) p_j(\mu)
\]

Where \( p_j(\mu) \) is probability of variable j to get value \( \mu \). Update rule:

\[
P_i^{k+1}(\lambda) = \frac{P_i^k(\lambda) S_i^k(\lambda)}{\sum_{\mu=1}^{m} P_i^k(\mu) S_i^k(\mu)}
\]

Implementation

Problem

The WCSP is generated as instance of class Problem. Each problem consist of n variables with domain size d and constrain matrix cons [][][][]. Cons[i][j][k][l] is represents \( r_{ij}(k,l) \). In order to generate constrain matrix we need two parameters p1 and p2:

- P1 – density: chance of constrain between variable i and j.
- P2 - tightness: chance of pair: chance of constrain between some values of variables i and j.

In addition the problem have max cost (price) parameter (mc) - constrain was random number between 1 and mc. mc was constant in this work, and set to 50. Zero constrain means no constrain.

BnB

BnB implemented as class BnB. It is implemented as described at reference 3. It returns the price of best assignment.
RL

RL implemented as class RelaxLabel, which gets the WCSP problem and number of iteration – k. It performs initialization - generation of random probability for each value for each variable (two dimensional matrix). Then for k iteration it calculates support and update probability as shown in above. Then assign values for variables with highest probability and calculate the total price.

Experiments and results.

In order to test RL WCSP was generate with same p1 and p2 parameter. P1 =p2 {0.6, 0.7, 0.8, 0.9}. It is know from previous research , which was conducted as part of “Constrain satisfaction” course, that below 0.6 we have low density constrain matrix – optimal solution almost always have price 0. RL perform quite well at those conditions at most of the time it finds the assignments with 0 total prices. That’s why focus was on the high density constrain matrix. Other important parameters of WCSP problem are number of variables – n and variable’s domain size-d. WCSP was generated as set of 50 problems with same n and d. Each problem solved:

1. BnB – to get the optimal price.
2. RL algorithm with 5 iterations. 10 runs in order to produce difference initial probabilities.
3. RL algorithm with 50 iterations. 10 runs in order to produce difference initial probabilities

On figure 2 we can see the results as average difference between BnB results and RL runs.

![Figure 2 Result for n=d=4. The x-axis is p1=p2. The y-axis is deference between final assignment of BnB and RL.](image)

Figure 3 shows the results for all problems with varying n and d. As we can see, size of the problem has devastating effect on RL. Generally speaking any local search heavily depends on start condition, and usually collapses to the local minimum. From figure 3 we can conclude, that for small problems RL will find optimal solution. In addition we can conclude from figure 3, that number of iteration will improve the solution. This conclusion lead us to next experiment: solving heavy WSCP with increasing number of iterations. The results of this experiment are shown on figure 4.

In order to test dependency on iteration number we choose most heavy problem: p1=p2 = {0.8, 0.9} and n=d=10. From figure 3 we can see that deference between 5 and 50 iterations are increasing with p1 and p2. From figure 4 we can see that increasing number of iteration does help to improve optimal solution, but at some point RL will stuck at local minimum. This can be concluded from figure 4: the deference between 100 and 500 iterations are neglected.
Conclusions

RL are local search algorithm, which can be successfully applied on WSCP problems in order to find optimal solution. The level of success depends on difficulty level of the problem itself:

- N and d parameters have great impact on optimization level of RL. But also it will increase the price of BnB run greatly. May be for some conditions it will be useful to use “cheap” and non-accurate method, than to stuck with BnB run.
- P1 and P2 parameters define the density of constrain matrix. When those parameters are above 0.6 there is no solution for CSP problem- no solution with price 0. Also we can say that number of iteration for stabilization of RL on local minimum is independent on p1 and p2 (figure 4).

In addition we can conclude that for heavy WSCP problems RL will not be able to find optimal solution even for high number of iterations. But average difference between optimal solution and RL solution will be in order of mc parameter.
Reference
1. Lecture notes “Computational and Biological Vision” by Ohad Ben-Shahar, 2014
2. Lecture notes “Constraints Processing” by Amnon Meisels, 2014