www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue -9 September, 2014 Page No. 8198-8201

Solving Knapsack Problem Using Constraint Programming

Vaishali Cooner

Department of CSE, U.I.E.T
Kurukshetra University, Kurukshetra, India
vaishalicooner@gmail.com

Abstract: ***A logical relation among several unknown variables is known as a constraint, where each variable in a given domain takes a value. The basic idea behind constraint programming framework is to model the problem as a set of variables with domains and a set of constraints.This problem has been studied since 1897 i.e. more than a century. In this paper we will consider a problem which will represent the basic features of constraint programming. We will here use constraint programming to solve this problem of knapsack. Our goal is to select items which have a greatest total value under certain conditions of limit on weight.***

Keywords: knapsack, constraint programming, optimization, logical relations

1. Introduction

Typically a knapsack problem involves some capacity (i.e. capacity of the weight which is carried) which has an upper limit and some different item sets that are to be chosen. Each item has a value and a cost which is presented in terms of capacity i.e. it has some weight [20].Goal here is to select item sets which have a greatest total value considering the limit on capacity.

This problem i.e. knapsack problem is easy to describe, but in case of large scale it is difficult to be worked out. The early work on knapsack has been started back in 1897 which show that a century has passed studying this problem of knapsack. Earlier this problem was carried out in a deterministic environment where the values and weights were positive crisp values [6]

.But due to lack of history data and technical difficulties and insufficient information, the data proved to be nondeterministic.

So, there were no such methods to solve this knapsack problem. Constraint programming made it easier to solve it. It is based on mathematical calculation so it is easier to understand and learn.

2. Methodology

Microsoft solver is considered as an awesome tool for solving constraint satisfaction problems. Defined as a set of objects they can be treated as mathematical problems where a number of constraints must be satisfied.

For this there are various ingredients:

- A set of one or more constraints, as your limitations.
- A set of one or more decisions to take, as your questions.
- A goal to reach, which is your ideal solution.

Using these 3 elements the best decision can be taken in order to meet the constraints.

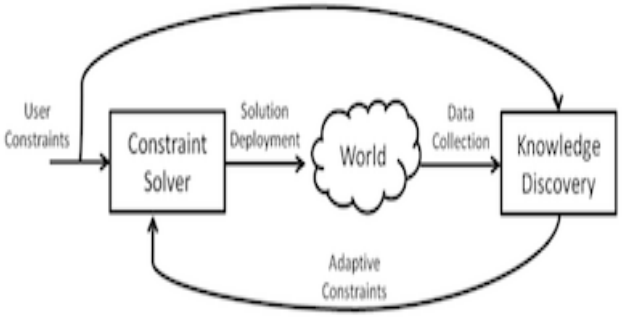

Figure 1: Icon Cycle

A few steps to follow are given:

Step 1: Deciding the decisions to take

Optimization is the first step that is to be defined. To enable the solver to do its job, what is required to be given is the domain of possible values. A integer is what we can only have for each of its decision.

Step 2: Defining the constraints
Define the constraints and add our constraints to the solver are the second next step to do.

Step 3: Determining goal
Finally what we want to do is, we need to tell the solver the goal. Optimized solution is the goal and we are to clearly

optimize our given constraints. The simple formula for our profit is defined and the solver is explained that we want to maximize it.

Final step: Wait for the final optimized answer.

Method Solve () is called finally which will help us in trying to solve our problems. An infeasible solution can always be there if given a problem which is impossible to solve. The output in this case can be "feasible", probably the solution is not optimal though the problem is solved.

Microsoft solver can be used to build and solve real optimization problems models. A modeling language is included for specifying optimization models. A .NET API is required for runtime model creation and analysis.

For modeling and mathematical optimization Microsoft solver foundation is a .NET solution. Real-world optimization models can be solved easily and can also be build easily by providing a .NET API using a solver foundation.

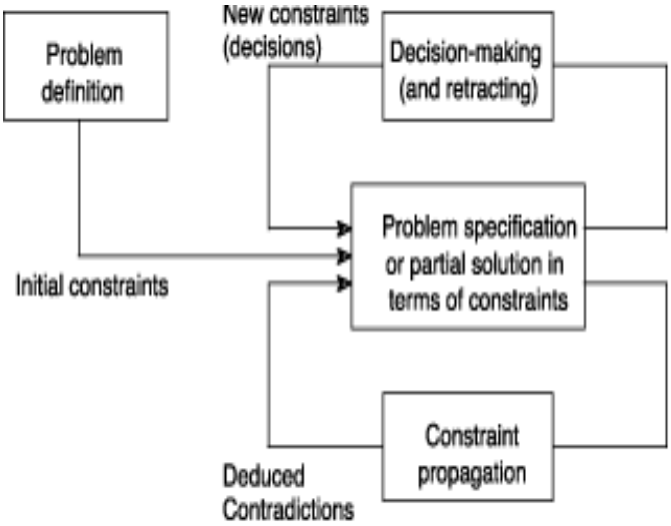

Figure 2: Behavior of a constraint programming system

3. Generalized Algorithm

The algorithm for knapsack problem using constraint programming is as follows:

- As terms of the problem, variables and constraints are defined.
- Constraint programming algorithms are specified
- Parameters are added as per problem
- As required by the problem constraints are added
- Decisions are added
- Goal is reached

4. Flow chart for the proposed work

A flowchart defines the steps to be followed for solving any problem using any method.

Here the problem is knapsack and method used to solve it is constraint programming.

Shown below is the flowchart for this problem:

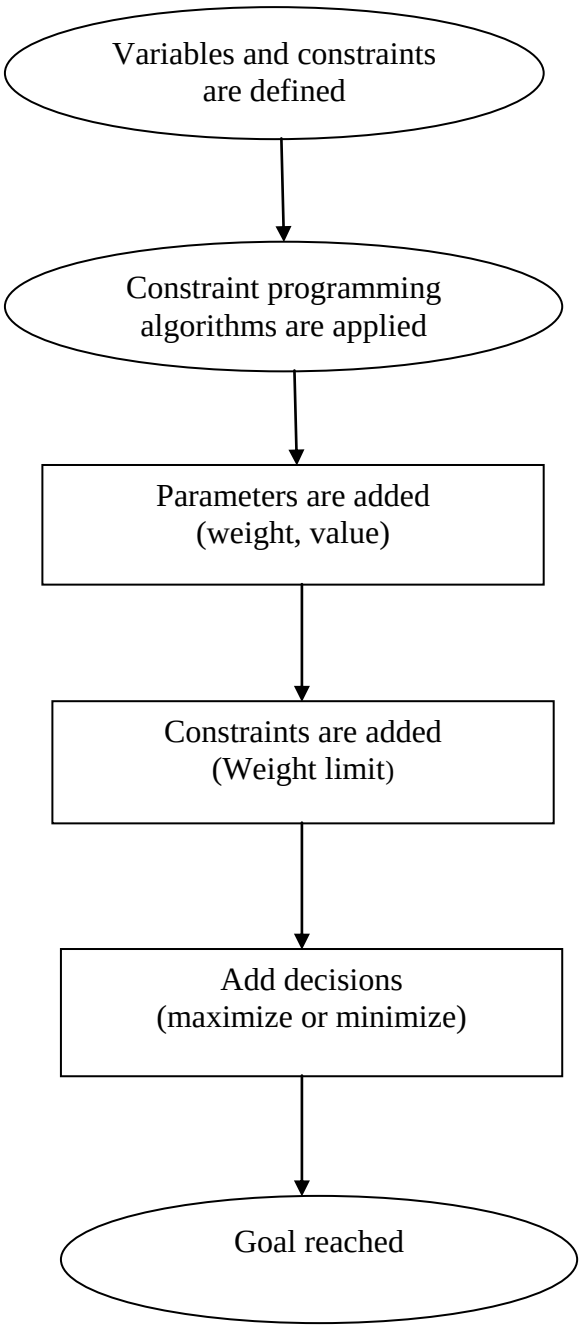

Figure 3: Flow chart for proposed work

5. Solving Knapsack Problem

We consider a problem where we have 8 different items which have a weight and a value. We are to take items considering value i.e. items of maximum value and total weight of whose should not exceed the maximum allowed weight.

The weight and value of the items is as follows:

Table 1

Item No.	Items	Value	Weight
1	Camera	15	2

2	Necklace	100	20
3	Vase	90	20
4	Picture	60	30
5	TV	40	40
6	Video	15	30
7	Chest	10	60
8	Brick	1	10

There are some features which are to be considered:

- Constants:

  WEIGHT=102

A constant is declared i.e. WEIGHT, and it is assigned a value 102. As 102 is an integer, WEIGHT is integer constant.

- Ranges:

  Item= 1…..8

It defines a set of range, i.e. integers from 1 to 8.

- Decisions:

  Decision (take)

It is to check whether the item is taken or not. It has a value 1 if the item is taken and a value 0 if the item is not taken.

- Parameters:

  Weight, value

- Constraint:

  Weight limit

- Goal:

  Maximize

```
Select New ItemDef With {
             .Name = eleItem.Attribute("name").Value,
             .Value =
Convert.ToDouble(eleItem.Element("value").Value,
provider),
             .Weight =
Convert.ToDouble(eleItem.Element("weight").Value,
provider)}

The parameters weight and value are set for the items
and the constraint weight limit is set. The goal is
set to achieve.

model.AddDecision(take)
model.AddParameters(_weight, value)
        model.AddConstraints("weightlimit",
model.Sum(model.ForEach(sItems, Function(item)
_weight(item) * take(item))) <= WEIGHT)
        model.AddGoal("valueGoal", GoalKind.Maximize,
model.Sum(model.ForEach(sItems, Function(item)
value(item) * take(item))))
```

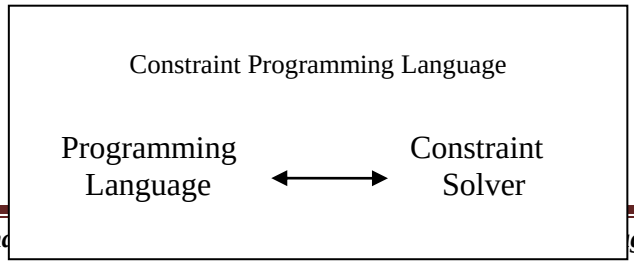

Figure 4

The solution to this problem as calculated in the form of 0 and 1 is:
take (1): 1
take (2): 1
take (3): 1
take (4): 1
take (5): 0
take (6): 1
take (7) :0
take (8): 0

So the items to be taken are: camera, necklace, vase, picture, video with values 15,100,90,60,15 respectively and weights: 2, 20, 20, 30, and 30.

6. Conclusion

Constraint programming is used here to solve knapsack problem, in this paper. An algorithm is developed which is efficient. The example discussed here is small, but it is expected that large problems can also be solved using this method. Items are selected based on values and weights, such that, maximum valued items are selected also considering the maximum given weight limit.

References

[1] H. Wang, G. Kochenberger, F. Glover, A computational study on the quadratic knapsack problem with multiple constraints, Computers & Operations Research, Vol. 39, No.1, 3-11, 2012.
[2] Arnaud Lallouet, Matthieu Lopez, Lionel Martin, Christel Vrain, "On Learning Constraint Problems," in ICTAI, 2010.
[3] Barry O'Sullivan, "Automated Modelling and Solving in Constraint Programming," in proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence(AAAI-10), 2010.
[4] C. Bessi`ere, R. Coletta, and T. Petit, "Learning implied global constraints," in IJCAI, 2007, pp. 44–49.
[5] Steven J. Miller," An Introduction to Linear Programming," in mathematics,2007.
[6] K. Schilling, Random knapsacks with many constraints, Discrete Applied Mathematics, Vol. 48, No. 2, 163-174, 1994.
[7] Wei Shen, Beibei Xu, Jiang-ping Huang, "An Improved Genetic Algorithm for 0-1 Knapsack Problems", Second International Conference on Networking and Distributed Computing 2011.
[8] Adrian Petcu," Recent Advances in Dynamic, Distributed Constraint Optimization," in infoscience, 2006.
[9] D. Chakrabarty, Y. Zhou, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In WWW2007, Workshop on Sponsored Search Auctions , 2007.
[10] C. Bessiere, J. Quinqueton, and G. Raymond, "Mining historical data to build constraint viewpoints," in Proceedings CP'06 Workshop on Modelling and Reformulation, 2006, pp. 1–16.
[11] Y. Liu, M. Ha, Expected value of function of uncertain variables, Journal of Uncertain Systems, Vol. 4, No. 3, 181-186, and 2010.

[12] Julia L. Higle," Stochastic Programming: Optimization When Uncertainty Matters," in operations research informs-New Orleans 2005, 2005.
[13] Chen Lin, "A Heuristic Genetic Algorithm Based on Schema Replacement for 0-1 knapsack Problem", Fourth International Conference on Genetic and Evolutionary Computing 2010.
[14] Y.C. Law, J.H.M. Lee," Model Induction: a New Source of CSP Model Redundancy," in AAAI, 2002.
[15] Hande Y. Benson, David F. Shanno, Robert J. Vanderbei," A Comparative Study of Large-Scale Nonlinear Optimization Algorithms," in NLP, 2002.
[16] Roman Bartak," Constraint-Based Scheduling: An Introduction for Newcomers," in SOFSEM, 2002.
[17] A. S. Anagun, and T. Sarac, ''Optimization performance of genetic algorithm for 0-1 knapsack problems using Taguchi method,'' Lect. Notes Comput. Sc., vol. 3982, pp. 678-687, 2006.
[18] H. Shih, Fuzzy approach to multilevel knapsack problems, Computers & Mathematics with Applications, Vol. 49, No. 7-8, 1157-1176, 2005.
[19] J.-F. Puget, "Constraint programming next challenge : Simplicity of use," in International Conference on Constraint Programming, ser. LNCS, M. Wallace, Ed., vol. 3258. Toronto, CA: Springer, 2004, pp. 5–8, invited paper.

[20] S. Martello, D. Pisinger, P. Toth, New Trend in exact algorithms for the 0-1 knapsack problem, European Journal of Operational Research 123 (2000) 325–332.

Author Profile

Vaishali Cooner received the B.tech degree in Computer science and engineering in 2012 and is doing M.tech(2012-2014) in Software engineering from U.I.E.T, Kurukshetra University, Kurukshetra. During M.tech she did research on constraint programming which is a part of mathematical engineering and has completed her research.