A PARTIAL REPRESENTATION TECHNIQUE FOR SUBSET SUMS

Nelson Maculan
Universidade Federal do Rio de Janeiro - UFRJ
COPPE Programa de Engenharia de Sistemas e Computação
Centro de Tecnologia, CxP 68.511
CEP 21.945-970 Rio de Janeiro - RJ
Nei Yoshihiro Soma
Instituto Tecnológico de Aeronáutica - ITA
Divisão de Ciência da Computação - IEC
Campo Montenegro - CEP 12.228-900
São José dos Campos, SP
Horácio Hideki Yanasse
Instituto Nacional de Pesquisas Espaciais - INPE
Laboratório Associado de Computação - LAC
Av. dos Astronautas, 1758 CxP 515 CEP 12.201-970
São José dos Campos, SP

Resumo: O Problema da Soma de Subconjuntos (SSP) pode ser posto como, "dados $n+1$ inteiros positivos, w_1, w_2, ..., w_{n-1}, w_n e W, encontre um subconjunto dentre os w_i's tal que sua soma seja exatamente igual a W". Apesar da simplicidade de sua formulação, sabe-se que este problema é NP-difícil, e que ele pode ser resolvido de maneira razoável quando os coeficientes são "pequenos", ou então muito "grandes". Todavia, quando os coeficientes estão entre estes dois extremos não há algoritmo algum conhecido na literatura. Propõe-se uma técnica de representação parcial dos coeficientes do problema que fornece uma alternativa possível para a resolução deste tipo de problemas.

Abstract: The Subset Sum Problem (SSP) can be posed as: "Given a set of $n+1$ positive integers, w_1, w_2, ..., w_{n-1}, w_n and W find a subset amidst the w_i's such that their sum is W". Despite its simple definition, this problem is a well known NP-Hard problem and when the coefficients are either "small" or "large" there are algorithms which solve it with a reasonable computational effort. However, no reference in the literature was found for problems where the coefficients are between these two extremes. To deal with this type of problems a partial representation technique approach is suggested.

Keywords: Integer Programming, Dynamic Programming, Combinatorial Optimization

1. Introduction

The Subset Sum Problem (SSP) can be stated as "Find a *Subset* amidst *n* positive integers $w_1,\ldots,w_{n-1}, w_n$ such that the *Sum* of this *Subset* is another positive integer *W*". It can also be posed as a Knapsack Problem, *i.e.* given a set of *n* items with weights w_i's and a Knapsack with capacity *W*, solve:

$$\begin{aligned} &\text{Maximize } z = \sum_{i=1}^{n} w_i x_i \\ &\text{Subject to :} \quad \sum_{i=1}^{n} w_i x_i = W \\ &x_i = 0 \text{ or } 1, \quad j \in \{1 \cdots n\} \\ &\text{where } x_j = \begin{cases} 1, \text{ if item j is selected} \\ \quad\quad \text{or} \\ 0, \text{ otherwise} \end{cases} \end{aligned} \qquad (1)$$

The Subset Sum Problem is a well known NP Hard member, *c.f.* Garey and Johnson (1979) and Martello and Toth (1990). There is, however, a strong relation between the maximum data size and both the space requirements and running time, *i.e.* if the coefficients are "small" then it can be solved in pseudopolynomial time by Dynamic Programming (*c.f.* Faaland (1973), Ahrens and Finke (1975)) or Hybrid Approaches (*c.f.* Martello and Toth (1984a), Yanasse and Soma (1987) and Yanasse and Soma (1994)). Typically, "small" are coefficients up to $q_{Max} \approx 9$ digits (where $q_{min} \leq \lfloor \log w_i \rfloor \leq q_{Max}$, for all i = 1, ..., n), since in addition to pseudopolynomial time algorithms, there are some *polynomial time* approximation schemes for the SSP with very good performances indeed *c.f.* Martello and Toth (1984b), Martello and Toth (1990) and Soma, Yanasse, Zinober and Harley (1995).

For "large" coefficients ($q_{min} \geq \text{Max}_{j=1,\ldots,n} \lfloor \log w_j \rfloor \geq 10\, n$) there is the Lagarias and Odlyzko (1983) algorithm which is a direct application of the L^3, *c.f.* Lenstra, Lenstra and Lovász (1982) and Lovász (1986). For the latter case, it is important to note, that the algorithm is not an exact one, but it finds a solution, if any, for *almost all* instances of SSP, provided that the coefficients are large positive integer numbers.

The authors are unaware of other works and references in the literature dealing with the SSP with intermediate size coefficients. This provides an indication, perhaps, that problems with these coefficients sizes, *i.e.* that are neither too "small" nor too "large" are the hardest ones to solve.

We suggest a partial representation technique which may solve these intermediate size coefficients. The main idea is to generate another SSP much easier than the original one, solve il and whenever a solution to this surrogate problem is found, test it in the original problem to know whether it is a solution or not. The surrogate SSP's are generated by using a similar idea as of *bit string slicing*. In the next section these ideas are further explored.

2. Mathematical Formulation

Let the constraint of (1) be decomposed into:

$$\begin{cases} \sum_{i=1}^{n} \overline{w}_i x_i = \overline{W} - \lambda & (2.1) \\ \sum_{i=1}^{n} \underline{w}_i x_i = \underline{W} + \Lambda\lambda & (2.2) \end{cases}$$

where,

$$\begin{cases} \overline{w}_i = (w_i - w_i \bmod \Lambda).\Lambda^{-1} & (3.1) \\ \underline{w}_i = w_i \bmod \Lambda & (3.2) \\ \overline{W} = (W - W \bmod \Lambda).\Lambda^{-1} & (3.3) \\ \underline{W} = W \bmod \Lambda & (3.4) \\ \lambda = 0, \cdots, n-1 & (3.5) \end{cases}$$

Λ is a positive integer number satisfying $\mathrm{Max.}_{j=1,\ldots,n}\{1, \lfloor \log w_j \rfloor - 9\} \leq \log \Lambda \leq \mathrm{Max.}_{j=1,\ldots,n}\{\lfloor \log w_j \rfloor\}$, and wherefore, it is straightforward to show that x is a solution to (1) *if and only if* it is a solution to (2.1) and (2.2), for some $\lambda = 0, \ldots, n-1$.

We suggest the following approach to solve (1).

Algorithm ***Partial Representation***

```
          1. Solve (2.2) by Dynamic Programming; for $\lambda = 0, \ldots, n-1$;
          2. Whenever a solution is found for a given stage and some $\lambda$ in the
above     interval, do a Backtrack to find $x$ (solution of (2.2)) and test it in (1),
              2.1 If $x$ is a solution to (1) then Halt (Optimal Solution),
```

2.2 Otherwise make a set all states of that given stage and λ as infeasible, (which is equivalent to a pruning in the incumbent tree) and go to step 1, *i.e.*continue with a new stage, or increase the value of λ.

3. If all stages and λ's have been considered and no solution was found, then problem (1) is infeasible.

Remarks:

1. The range of λ can assume in (2) (see (3.5)), probably can be tightened. Better estimates can be obtained in polynomial time by using Dantzig's greedy algorithm (or Martello and Toth quadratic greedy search, *c.f.* Martello and Toth (1990)). If the range of λ is made tighter (in the better case it will be 1) then the computational effort decreases sharply.
2. There exists a trade-off between the value of Λ to be chosen and the number of solutions in (2.2), *i.e.* if $\Lambda=1$ then the number of solutions in (2.2) will be almost surely exponential, but the space requirement will be pnlynomial and if $\Lambda = 10^{\text{Max.}\{\lfloor \log w_j \rfloor\}}$ then the reverse case occurs, so it seems reasnnable that the better values of Λ cannot occur in the extremes of the interval.
3. The *pruning* in the incumbent tree is made *before* taking into account a new variable, since it will diminish the number of "misleading" feasible solutions to be tested.
4. Depending upon the choice of Λ some of the w_j's can be zero, which seems to be an additional difficulty. This case, however, can be handled easily by splitting the set of coefficients into two subproblems, *i.e.* those with the $w_j = 0$ as subproblem (I) and those with $w_j \neq 0$ as subproblem (II). Solve subproblem (II) with the algorithm given above and subproblem (I) by Dynamic Programming and then do a merge between the solutions of the two subproblems.

3. Conclusions

Some limited computational tests evaluating the performance nf this proposed approach will be presented.

The suggested method can be used as an alternative for solving "*hard*" Subset Sum Problems, although the algorithm is *pseudopolynomial* in time, due to size of the coefficients for practical purposes its worst running time case can be considered exponential.

Another point which seems to deserve some further study is the study of *Aggregative Methods* (which was in vogue during the 70's, *c.f.* Salkin and Mathur (1989)) in conjunction with the theory of cutting planes. It seems that the main reason for the wane of the former, came with the sizes of the resulting coefficients.

Finally, it seems also interesting to try this approach in a distributed or parallel systems.

4. Bibliography

J.H. Ahrens and G. Finke (1975). Merging and Sorting applied to the 0-1 knapsack problem. *Operations Research* **23**, 1099-1109.

B. Faaland (1973). Solution of the value-independent knapsack problem by partitioning. *Operations Research* **21**, 332-337.

M.R. Garey and D.S. Johnson (1979). *Computers and Intractability: A guide to the theory of NP-Completeness*, Freeman, San Francisco.

J.C. Lagarias and A.M. Odlyzko (1983). Solving low density subset sum problems. *Proc. 24th Annual IEEE Symposium Foundation of Computer Science*, 1-10.

A.K. Lenstra, H.W. Lenstra and L. Lovász (1982). Factoring polynomials with rational coefficients, *Math. Ann.*261, 515-534.

L. Lovász (1986) *An algorithmic theory of numbers, graphs and convexity*, CBMS-NSF Reg. Conf. series in Appl. Math. SIAM, Philadelphia.

S. Martello and P. Toth (1984a). A mixture of dynamic programming and branch-and-bound for the subset sum problem. *Management Science* 30, 765-771.

S. Martello and P. Toth (1984b).Worst-case analysis of greedy algorithms for the subset sum problem. *Mathematical Programming* **28**, 198-205.

S. Martello and P. Toth (1990). *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester.

H.M. Salkin and K. Mathur (1989). *Foundations of Integer Programming*, Elsevier, New York.

N.Y Soma, H.H. Yanasse, A.S.I. Zinober and P.J. Harley (1995) A polynomial approximation scheme for the subset sum problem, *Discrete Applied Mathematics* **57**, 243-253.

H.H Yanasse and N.Y. Soma (1987) A pseudopolynomial algorithm for the 0-1 value independent knapsack problem. **In:** Anais do XX SBPO, Salvador.

H.H. Yanasse and N.Y. Soma (1994) A new bound for a linear diophantine equation problem, Investigación Operativa, 3 (**1**).