A logo with a lion and building

Description automatically generated with medium confidence

POLITECNICO DI TORINO

DEPARTMENT OF CONTROL & COMPUTER ENGINEERING

*COMPUTATIONAL INTELLIGENCE*

**Class Activity\_02**

**Mathematical Reduction of Set Covering to Linear Programming**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Matricula** | **GitHub** | **Date**  **submission** |
| Hossein Khodadadi | 313884 | [Link\_1](https://github.com/HOSSENkhodadadi/Computational_Intelligence/tree/main/Class_Activity_01) | 24/10/2023 |
| Abolfazl Javidian | 314441 | [Link\_2](https://github.com/Abolfazl-Javidian/Computational-Intelligence/tree/main/Class_Activity_01) | 24/10/2023 |

**https://optimization.cbe.cornell.edu/index.php?title=Set\_covering\_problem**

Fall 2023/2024

**Mathematical Reduction of Set Covering to Linear Programming**

We begin by formulating the set cover problem as an Integer Linear Programming problem. Given an input (U, S1, . . . , Sn) of the set cover problem, we introduce a variable xi for every set Si, with the intended meaning that xi = 1 when Si is selected, and xi = 0 otherwise. We can express the set cover problem as the following integer linear program:

minimize

subject to

i:v∈Si xi ≥ 1 ∀v ∈ U

xi ≤ 1 ∀i ∈ {1, . . . , n}

xi ∈ N ∀i ∈ {1, . . . , n}

(1)

From which we derive the linear programming relaxation

minimize

subject to

i:v∈Si xi ≥ 1 ∀v ∈ U

xi ≤ 1 ∀i ∈ {1, . . . , n}

xi ≥ 0 ∀i ∈ {1, . . . , n}

More generally, it is interesting to consider a *weighted* version of set cover, in which we are given the set *U* , the collection of sets *S*1*, . . . , Sn*, and also a *weight wi* for every set. We want to find a sub-collection of *minimal total weight* whose union is *U* , that

S Σ

is, we want to find *I* such that *i*∈*I Si* = *U* , and such that *i*∈*I wi* is minimized. The unweighted problem corresponds to the case in which all weights *wi* equal 1.

The ILP and LP formulation of the unweighted problem can easily be generalized to the weighted case: just change the objective function from *i xi* to *i wixi*.

Σ Σ

minimize Σ*n*

*i*=1

*wixi*

subject to

Σ

*i*:*v*∈*Si xi* ≥ 1 ∀*v* ∈ *U*

*xi* ≤ 1 ∀*i* ∈ {1*, . . . , n*}

*xi* ≥ 0 ∀*i* ∈ {1*, . . . , n*}

(3)

Suppose now that we solve the linear programming relaxation ([3](#_bookmark0)), and we find an optimal fractional solution **x**∗ to the relaxation, that is, we are given a number *xi*∗ in the range [0*,* 1] for every set *Si*. Unlike the case of vertex cover, we cannot round the *xi*∗ to the nearest integer, because if an element *u* belongs to 100 sets, it could be that *xi*∗ = 1*/*100 for each of those sets, and we would be rounding all those numbers to zero, leaving the element *u* not covered. If we knew that every element *u* belongs

to at most *k* sets, then we could round the numbers 1*/k* to 1, and the numbers

≥

*<* 1*/k* to zero. This would give a feasible cover, and we could prove that we achieve a *k*-approximation. Unfortunately, *k* could be very large, even *n/*2, while we are trying to prove an approximation guarantee that is never worse than *O*(log *n*).

Maybe the next most natural approach after rounding the *x*∗*i* to the nearest integer is to think of each *xi*∗ as a *probability*, and we can think of the solution **x**∗ as describing a probability distribution over ways of choosing some of the subsets *S*1*, . . . , Sn*, in which we choose *S*1 with probability *x*1∗, *S*2 with probability *x*∗2, and so on.

Algorithm *RandomPick*

* + Input: values (*x*1*, . . . , xn*) feasible for ([3](#_bookmark0))
  + *I* := ∅
  + for *i* = 1 to *n*

**–** with probability *xi*, assign *I* := *I* ∪{*i*}, otherwise do nothing

* + return *I*

Σ

Using this probabilistic process, the expected cost of the sets that we pick is *i wix*∗*i* , which is the same as the cost of **x**∗ in the linear programming problem, and is at most the optimum of the set cover problem. Unfortunately, the solution that we construct

could have a high probability of missing some of the elements.

Indeed, consider the probabilistic process “from the perspective” of an element *u*. The element *u* belongs to some of the subsets, let’s say for simplicity the first *k* sets *S*1*, . . . , Sk*. As long as we select at least one of *S*1*, . . . , Sk*, then we are going to cover

*u*. We select *Si* with probability *x*∗*i* and we know that *x*∗1 + *x*∗*k* 1; what is the probability that *u* is covered? It is definitely not 1, as we can see by thinking of the case that *u* belongs to only two sets, and that each set has an associated *x*∗*i* equal to 1*/*2; in such a case *u* is covered with probability 3*/*4. This is not too bad, but

· · · ≥

maybe there are *n/*10 elements like that, each having probability 1*/*4 of remaining uncovered, so that we would expect *n/*40 uncovered elements on average. In some examples, Ω(*n*) elements would remain uncovered with probability 1 2−Ω(*n*). How do we deal with the uncovered elements?

−

First of all, let us see that every element has a reasonably good probability of being covered.