University of Ottawa/
Université d'Ottawa

**CSI4105 Type 1 Project**

# Weighted Set Covering Problem Research Proposal

| Ali Mowazi | Jeremiah O'Neil | Mark Kasun |
|---|---|---|
| 5879953 | 6498391 | 3806554 |

12th February 2016

## 1 Introduction to Set Cover

Set Cover in decision form is a NP- Complete problem that has a few different forms such as Vertex Cover or Weighted Set Cover. Our team will be researching and covering Weighted Set Cover. Set Cover is best explained as minimum number of subsets needed that when taken the union of these subsets they together cover all elements in the given set. In decision form for weighted set cover as defined by Yair Bartal (2005) we are given a Set $u$ of $n$ elements, a collection of subsets $S_1, S_2, \dots, S_m$ of subsets with weight $wt(i)$ for each subset and a number $k$, then we are asked if there is a collection of at most $k$ subsets with at most weight of $wt(L)$. Richard M. Karp (1972) showed in "Reducibility Among Combinatorial Problems" that there is a polynomial time reduction from SAT to Set Cover, proving it is NP-Complete. There are a few well known attempts at solving set cover such as the greedy algorithm or through linear programming. Set Cover has many real world applications, some are fairly famous applications. IBM realized it had 5000 known computer viruses that had 9000 substrings, they were able to find that only 180 needed subsets to cover the 5000 viruses (David P. Williamson, 1998). So instead of searching for 5000 viruses they only need to search for 180 subsets of consecutive bytes that would cover the 5000 viruses. We will go on to show two methods to tackle the Weighted Set Cover problem. We will have one greedy algorithm and algorithm using simulated annealing.

## 2 Test Data

The OR-Library at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html` includes numerous test data sets for set cover which we can use to test our algorithms. Originally, we were going to look at unweighted set cover but the libraries we found all dealt with weighted set cover and our the algorithms we wished to look at were easily adaptable to deal with it. The library includes references to where we can find optimal solutions for the test data which will greatly enhance our work.

## 3 Greedy Set Cover

For the project, we will be looking at the greedy algorithm for solving general set cover problems. The greedy algorithm is an approximation algorithm with a guaranteed performance compared with the optimal solution to the problem. Unlike most approximation algorithms, the guarantee is not a constant multiple of the optimal but instead a it is a function multiple of the optimal based on the size of the input for the instance. The greedy algorithm is guaranteed provide a solution that is less than or equal to $\alpha \cdot OPT(I)$ where $\alpha$ is given by

$$\alpha = H_k = \sum_{i=1}^{k} \frac{1}{i} \leq 1 + \log(k)$$

where $k$ is the size of the largest subset and $H_k$ is the $k$th partial sum of the harmonic series. Since $k$ will always be less than $n$, we can re-frame $\alpha$ in terms of $n$ if so desired.

The implementation of the algorithm is fairly simple. Initialize an array that holds the cost effectiveness for each subset where cost effectiveness is the cost of the subset divided by the number of elements in the subset. Take the set with the lowest cost effectiveness and consider all elements in the subset covered. Update the cost effectiveness of all sets excluding the elements covered by the previous set. Repeat the process until all elements in the set are covered. The subsets chosen through the process are the result of the algorithm. The algorithm works on unweighted set cover where the weights of each subset is simply 1. The algorithm has a worst-case running time of $O(N \log(N))$ where $N$ is the actual size of the input including all subsets.

The guarantee of $H_k \cdot OPT(I)$ seems weak compared to algorithms for other NP-Hard problems where there are 2-approximation guarantees. The harmonic series quickly increases past 2, passing it for $k > 4$. However, it has been shown that the efficiency of the greedy algorithm for general set cover problems cannot be improved to a constant nor to anything significantly more efficient than the greedy algorithm (Pusztai, 2008).

## 4 Simulated Annealing

Simulated annealing (SA) is a stochastic strategy to guide local-search heuristic algorithms towards the global optimum in problems of combinatorial optimization. In particular, it guides local-search heuristics – which perturb a given solution to produce and evaluate solutions in its neighbourhood with respect to the perturbation, select a better solution, and repeat, eventually converging to a solution which is locally optimum – towards the global optimum by prescribing random acceptance of solutions which are less optimal than the current solution at a decreasing rate which depends on the relative cost of the solutions. These suboptimal solutions enable the algorithm to "jump" out of local optima and explore more of the problem domain, and SA's acceptance probability is tightened according to a schedule which is designed such that the algorithm converges on the global optimum with high probability; or an acceptable optimum at a high rate, as desired.

SA was introduced by Kirkpatrick, Gelatt, and Vecchi (1983), inspired by the Metropolis-Hastings algorithm (Metropolis et al. 1953) of statistical mechanics, and modelled after the physical process of annealing. The essence of SA is the analogy between combinatorial optimization and the problem of determining the lowest-energy ground state configuration of a physical system; and though physically motivated, SA is purely a mathematical method and is almost universally applicable to combinatorial optimization problems.

There are three key components to a SA algorithm: the heuristic for the generation of new solutions from the current one, the probability of accepting a new solution based on its cost, and the tightening schedule (the "annealing schedule") of the acceptence probability. The first is problem-specific, and should aim for solutions with cost similar to the current solution in order to provide a smooth cost landscape; which eases convergence to the optimum, and is an important requirement of the physical and mathematical justification of the method (Gendreau and Potvin 2010 give an excellent review of the latter). A simple and effective approach for the weighted set-covering problem is to remove sets at random, construct a trial solution by the greedy heuristic, and then cull any redundant columns; variations of this approach have been explored and validated by Brusco, Jacobs, and Thompson n.d. The probability of accepting a solution in SA is prescribed by the Boltzmann factor of statistical mechanics – an exponential in the relative cost of solutions scaled by the "temperature" – which is mathematically motivated by the Principle of Maximum Entropy and central to the Metropolis-Hastings algorithm; we'll probably leave this alone. Mézard and Montanari 2009 describe the possibility of extended definitions, but we've seen no such applications to the set covering problem. The annealing schedule provides the most room to play; it describes the overall rate, and manner – linear, logarithmic, variant, *etc.*– of change of the temperature, which affects both performance and running time in ways which can in general only be quantified and optimized through extensive benchmarking. We are particularly interested in implementing the Thermodynamic Simulated Annealing (TSA) method due Vicente, Lanchares, and Hermida 2003, which dynamically determines the schedule according to physical law; a stretch of the physical analogy of SA which even allows the temperature to increase under certain conditions, and is largely self-tuning. The set covering problem would be a novel application of TSA.

Other important design considerations include initial solution selection and termination criteria. These topic will be explored as the project develops.

## References

Brusco, M.J., L.W. Jacobs, and G.M. Thompson. "A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems". In: *Annals of Operations Research* 86, pp. 611–627. ISSN: 1572-9338. DOI: `10.1023/A:1018900128545`. URL: `http://dx.doi.org/10.1023/A:1018900128545`.

Gendreau, M. and J.Y. Potvin (2010). *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer US. ISBN: 9781441916655. URL: `https://books.google.ca/books?id=xMTS5dyDhwMC`.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). "Optimization by Simulated Annealing". In: *Science* 220.4598, pp. 671–680. ISSN: 0036-8075. DOI: `10.1126/science.220.4598.671`. URL: `http://science.sciencemag.org/content/220/4598/671`.

Metropolis, Nicholas et al. (1953). "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. DOI: `http://dx.doi.org/10.1063/1.1699114`. URL: `http://scitation.aip.org/content/aip/journal/jcp/21/6/10.1063/1.1699114`.

Mézard, M. and A. Montanari (2009). *Information, Physics, and Computation*. Oxford Graduate Texts. OUP Oxford. ISBN: 9780198570837. URL: `https://books.google.ca/books?id=jhCM7i0a6UUC`.

Vicente, Juan de, Juan Lanchares, and Roman Hermida (2003). "Placement by thermodynamic simulated annealing". In: *Physics Letters A* 317.5-6, pp. 415–23. ISSN: 0375-9601. DOI: `http://dx.doi.org/10.1016/j.physleta.2003.08.070`. URL: `http://www.sciencedirect.com/science/article/pii/S0375960103013653`.

David P. Williamson(1998) Lecture Notes on Approximation Algorithms Retrieved from people.orie.cornell.edu/dpw/cornell.ps

Richard M. Karp (1972). "Reducibility Among Combinatorial Problems"

Pusztai, Pál. "An Application of the Greedy Heuristic of Set Cover to Traffic Checks." Central European Journal of Operations Research 16.4 (2008): 407-14. Print.

Yair Bartal (2005) Advanced Algorithms. Lecture retrieved from `http://www.cs.huji.ac.il/course/2005/algo2/scribes/lecture2.pdf`

Young, Neal E. "Greedy Set-Cover Algorithms." Encyclopedia of Algorithms. Ed. Ming-Yang Kao. New York, NY: Springer, 2008. 379-81.