A faster parameterized algorithm for set packing

Ioannis Koutis
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
`ioannis.koutis@cs.cmu.edu`

December 14, 2004

Abstract

We present an efficient parameterized algorithm for the (k,t)-set packing problem, in which we are looking for a collection of k disjoint sets whose union consists of t elements. The complexity of the algorithm is $O(2^{O(t)}nN\log N)$. For the special case of sets of bounded size, this improves the $O((ck)^k n)$ algorithm of Jia et al. [2004, J. Algorithms, 50(1)].

1 Introduction

We study the parameterized complexity of the set packing problem. An instance of the problem asks whether a given collection C of n sets contains k mutually disjoint sets. It is known that, with k being the parameter, the problem is W[1]-complete [2]. This means that we probably cannot expect an algorithm of complexity polynomial in n, and proportional to $f(k)$, for any function f.

However, even the m-set packing problem where the size of the sets in C is bounded by $m \geq 3$ is NP-complete [3]. In view of this, Jia et al. [4] studied its parameterized complexity. They showed that the problem is fixed parameter tractable by giving an algorithm that runs in time $O\left(m^k\left(g(m,k)^{mk} + k^2m^2n\right)\right)$ where $g(m,k)$ is linear in mk.

We consider a more general problem, the (k,t)-set packing problem, in which we are looking for a collection of k disjoint sets whose union consists of t elements. We present an $O(e^t(tnN + t^2 4^t \log k))$ randomized algorithm and an $O(2^{O(t)}nN\log N)$ deterministic algorithm, where N is the total number of distinct elements of the sets in C. For the special case of the m-set packing problem, where $t = mk$, this yields an improvement over the algorithm in [4], with respect to the parameters. For example, for fixed m and $k = O(\log n)$ our algorithm is polynomial, whereas the algorithm of [4] is not. We note however that our algorithm is slower for a small range of the parameters.

In Section 2 we give an $O(kn^2N^2 2^N)$ algorithm for the general set packing, via an algebraic formulation of the problem. We do so, despite the fact that the algorithm can be stated purely in set-theoretic language, because apart from simplifying the presentation, the algebraization consists a more general approach which may be of independent interest. Then, in Section 3, we show how to extend the basic ideas of this algorithm, to obtain the randomized algorithm for the (k,t)-set packing problem. Finally, we derandomize the algorithm by adapting the color coding method of Alon et al.[1].

2 Algebraization of the Set Packing problem

2.1 The decision algorithm

Let $C = \{S_1, \ldots, S_n\}$, $U = \bigcup S_i$ and $N = |U|$. We assign variables x_i, $i = 1, \ldots, N$, to the elements of U. For each set $S \in C$ let $f(S)$ be the product of the variables corresponding to its elements. We will refer to $f(S)$ as the representation of S. Consider the multivariate polynomial

$$P_k = \left(\sum_{S \in C} f(S) \right)^k$$

If we expand P_k to a sum of terms and ignore their coefficients, we get one monomial per each different selection of k sets from C. Each of these monomials is the product of the representations of the corresponding sets. If a term contains two intersecting sets, then it is a multiple of x_i^2 for some i. On the other hand, if there are k disjoint sets with total size t, the term corresponding to the products of their representations is a multilinear monomial of degree t, in other words it is a product of t distinct variables.

Let us give a small example to illustrate the concept. Let $S_1 = \{1,2\}, S_2 = \{2,3\}, S_3 = \{3\}$. We let $1 \to x_1, 2 \to x_2, 3 \to x_3$, so that $f(S_1) = x_1x_2, f(S_2) = x_2x_3, f(S_3) = x_3$. Then,

$$\begin{aligned} P_2 &= (x_1x_2 + x_2x_3 + x_3)^2 \\ &= x_1^2x_2^2 + x_2^2x_3^2 + x_3^2 + 2x_1x_2^2x_3 + 2x_2x_3^2 + 2x_1x_2x_3 \end{aligned}$$

It can be seen that in P_2 there is only one multilinear monomial ($x_1x_2x_3$), and that it corresponds to the disjoint sets S_1, S_3.

For any polynomial Q let $M(Q)$ be the sum of the multilinear monomials of Q. That is, $M(Q)$ is a multilinear polynomial with every coefficient equal to 1. From the above discussion it follows that we can decide an instance of the problem by computing $M(P_k)$. Concretely, there is collection of k disjoint subsets in C if and only if $M(P_k) \neq 0$. Now, observe that for any Q_1, Q_2 we have $M(Q_1Q_2) = M(Q_1M(Q_2))$. By using this fact, it follows by induction that we can compute $M(P_k)$ in k steps:

$$M(P_i) = M(M(P_{i-1})P_1) \quad \text{for} \quad i = 2, \ldots, k$$

At each step we perform one multiplication and keep the multilinear terms. Since $M(P_i)$ contains only multilinear terms in N variables, it has at most 2^N terms, and each term can be fully determined with $O(N)$ bits. Also, P_1 has n terms. Hence, the multiplication can be performed in time $O(nN^22^N)$. The product $M(P_i)P_1$ is a sum of at most $n2^N$ monomials of size $O(N)$. Computing $M(M(P_i)P_1)$ can be done by sorting (with respect to some natural order) the terms of $M(P_i)P_1$, in time $O(Nn2^N \log(n2^N)) = O(nN^22^N)$. Since we do k multiplications, the overall complexity for computing $M(P_k)$ is bounded above by $O(knN^22^N)$.

2.2 Constructing a set packing

Now suppose we have computed $M(P_k) \neq 0$. Fix any term w of $M(P_k)$. We would like to find a collection R of k subsets such that the product of their representations is w. Consider the following algorithm:

```
1. $R = \{\}$
2. for $i := 1$ to $n$
3.     if $w \in \left(\sum_{S \in C - S_i} f(S)\right)^k$ then $C := C - S_i$ else $R := R \bigcup S_i$
4.     if $|R| = k$ then return $R$
```

At any point of the algorithm, 2^C contains a set of collections of k disjoint sets, say $\mathcal{C} = \{C_1, \ldots, C_j\}$, such that, for each i, the product of the representations of the sets in C_i is w. Step 3 ensures that for any set $S \in R$ we have $S \in \bigcap_{i=1}^{j} C_i$, and thus $R \subseteq \bigcap_{i=1}^{j} C_i$. When the algorithm terminates, $|R| = k$, and since $|C_i| = k$ there is only one collection in $\mathcal{C}$, and it is identical to R. Finally, note that the complexity of Steps 3 and 4 is dominated by the computation of $M(P_k)$. As shown in the previous section, this can be done in time $O(knN^2 2^N)$. Hence, the total complexity for computing R is bounded above by $O(kn^2 N^2 2^N)$.

3 The algorithm for (k,t)-set packing

3.1 The randomized algorithm

Let us identify the elements of U with variable names. We take a random mapping $M : U = [x_1, \ldots, x_N] \to U' = [y_1, \ldots, y_t]$. We construct the multi-set S_i' from S_i by substituting the elements in S_i according to M, and we let $C' = \{S_1', \ldots, S_n'\}$. If a multi-set S_i' has multiple copies of the same element we remove it from C'. If a number of sets from C map to the same set in C', we keep one of the copies. In this way we can think of C' as a sub-collection of at most 2^t sets from C. It is easy to see that if $S_i \bigcap S_j \neq \varnothing$ then $S_i' \bigcap S_j' \neq \varnothing$. It follows that if C does not contain a collection of k disjoint sets, then the same holds for C'. On the other hand, if C contains k disjoint subsets with a total of t elements, then the same holds for C' if the t critical elements get different names under M. This will happen with probability $t!/t^t > e^{-t}$.

It is not hard to see that for a given mapping M, an instance C' can be constructed from C in time $O(tnN)$. Now, we concentrate on C' and work with multilinear polynomials on t variables. For any two such polynomials, Q_1, Q_2, computing the product $Q_1 Q_2$ takes time $O((2^t t)^2)$. $Q_1 Q_2$ is a sum of at most 4^t monomials of size $O(t)$ and thus, $M(Q_1 Q_2)$ can be computed in time $O(t^2 4^t)$ by sorting the terms of $Q_1 Q_2$. By adapting the algorithm of Section 2, it follows that the instance C' can be decided in time at most $O(k(2^t t)^2)$. We can get an improvement to $O(t^2 4^t \log k)$ by using a standard technique for computing powers. To decide C' with constant probability we must try $O(e^t)$ random mappings. This takes time $O(e^t(tnN + t^2 4^t \log k))$. Once the right mapping is found, the k disjoint subsets in C' can be constructed as in Section 2 in time $O(2^t t^2 4^t \log k)$. By using the correspondence of sets in C' with sets in C we can recover k disjoint subsets in C. Thus, the complexity of the randomized algorithm is $O(e^t(tnN + t^2 4^t \log k))$.

3.2 Derandomization

To derandomize the algorithm we need a list $\mathcal{M}$ of mappings $M : U \to U'$ such that for every subset $V \subset U$ with $|V| = t$, there is a mapping $M \in \mathcal{M}$ such that every element of V is mapped (under M) to a different element in U'. This can be accomplished with a t-perfect family of hash functions. Such a family exists, it has size $2^{O(t)} \log N$ and can be constructed deterministically in time $O(2^{O(t)} \log N)$ [1]. It follows that a collection of k disjoint sets with a total of t elements can be found in time $O(2^{O(t)} nN \log N)$.

4 Acknowledgements

I would like to thank Gary L. Miller for his support, and Benoît Hudson for suggestions that helped in the improvement of the paper. I also thank the anonymous referees for their helpful comments and suggestions.

References

[1] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[2] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[4] Weijia Jia, Chuanlin Zhang, and Jianer Chen. An efficient parameterized algorithm for m-set packing. *J. Algorithms*, 50(1):106–117, 2004.