An Alternative Problem for Backtracking and Bounding

Timothy J. Rolfe
Computer Science Department
Eastern Washington University
Cheney, Washington 99004-2412 USA
Timothy.Rolfe@ewu.edu

Paul W. Purdom, Jr.
Computer Science Department
Indiana University, Bloomington
Bloomington, Indiana 47405-7104 USA
pwp@cs.indiana.edu

Abstract

One of the programming problems in the 2002 Pacific Northwest regional ACM ICPC contest provides a new way to teach backtracking and also provides a very powerful example of a forward-looking bounding function. This article presents the problem, the bounding function, and timing information of implementations with and without the bounding function. It also provides the URL for access to the programs themselves.

Keywords: Backtracking, bounding function, permutation

1. The Problem

In 1985, Dean Clark proposed a problem in the *American Mathematical Monthly* regarding permutations of numbers on the face of a clock [1]. This was picked up by Martin Gardner and presented to a wider audience in his puzzle column in *Isaac Asimov's Science Fiction Magazine* for August of 1986. Here is his statement of the problem.

> Now for a curious little combinatorial puzzle involving the twelve numbers on the face of a clock. Can you rearrange the numbers (keeping them in a circle) so no triplet of adjacent numbers has a sum higher than 21? This is the smallest value that the highest sum of a triplet can have.
>
> I know of no procedure for finding such a permutation, but there must be a way to write a computer program that will print all such permutations in a reasonable time [2].

This prompted one of the authors (Timothy Rolfe) to do exactly that, reporting the generation of such a computer program (using backtracking to bound the time required) in *Mathematics and Computer Education* [3], and more recently (thanks to its inclusion in the 2002 Pacific Northwest regional contest [4] for the ACM International Collegiate Programming Contest (ICPC) [5]) in *Dr. Dobb's Journal* [6].

In response to that article, the other author (Paul Purdom) provided a forward bound that greatly reduces the time required to solve the problem, and especially the time required to discover that there *are* no solutions for a maximum triplet sum less than 21.

2. Backtracking and Bounding

The problem may be useful in teaching backtracking, giving the Eight Queens a royal rest. It shows both the power and the limitation of a pure backtracking approach. One may view the problem as the examination of all permutations, counting up the ones that meet the condition. As a contest problem, this led some contestants to blithely invoke the C++ Standard Template Library function *next_permutation* — and exceed the time limit when they ran their program with the judges' input data. Recursive algorithms easily generate permutations and the backtracking can be embedded within the permutation algorithm itself. Such an implementation can detect the point at which a triplet is generated whose sum exceeds the limit and prune the decision tree there. This *greatly* reduces the processing required to find all valid clock faces. This implementation, though, has the disadvantage of continuing down the decision tree for a branch that *cannot* generate a solution because the numbers at the front of the permutation are too small (and thus those at the end are too large to produce small sums).

Suppose one assigned numbers to the clock face for positions 0 to k, but has not yet assigned numbers for positions k+1 to N–1. Here is the situation:

$X[k\text{-}1]$	$X[k]$	$X[k\text{+}1]$	...	$X[N\text{-}1]$	$X[0]$	$X[1]$
known	known	unknown	...	unknown	known	known

The sum of each three numbers should be no more than MaxSum, and there are N–k+1 groups of unknown sums-of-three.

Let R be the sum of the numbers not yet assigned. Then $X[k\text{–}1]$ appears in one group-of-three, $X[k]$ appears in two groups, each $X[i]$ contributing to R appears in three groups, $X[0]$ appears in two groups, and $X[1]$ appears in one group. Thus, it is possible to obtain a solution from a given front end to the permutation if the following condition holds:

$(N–k+1)$ MaxSum >= $X[k–1] + 2\ X[k] + 3\ R + 2\ X[0] + X[1]$

This bound is especially effective in problems for which there are few valid solutions.

The bounded backtracking implementation was programmed both in C and in Java [7]. The following table shows the results of running the Java implementation of the backtracking algorithm both without and with the forward bound, and capturing both the number of function calls and the elapsed time. In each case, the MaxSum used is the smallest one that has any valid permutations. Note that each valid permutation has a mirror image that is also a valid permutation. The program only counts unique permutations, discarding those equivalent as mirror images.

The program was run on Xeon processors in DELL quad-processor computers under the DELL-installed Red Hat Linux operating system. The program was compiled and run in the Java™ 2 Runtime Environment, Standard Edition (build 1.4.2_01-b06). While the Xeon is rated as 3.0 GHz, that is the result of hyperthreading two 1.5 GHz processors, so that the Linux system sees those two processors as running at 1.5 GHz.

Acknowledgements

The programs were run on computers acquired as part of the "Technology Initiative for the New Economy" grant by the federal government to Eastern Washington University that, among other things, provided a parallel and distributed processing resource—which these computers do admirably well! Each DELL is effectively an eight-processor SMP, so that among the five machines there are 40 processors available for distributed processing.

N	MaxSum	# Solns	Calls w/o	Calls w	Sec w/o	Sec w
12	21	261	187364	17842	0.02825	0.011
13	23	2842	1731873	226800	0.18325	0.092
14	24	144	5194742	117625	0.569	0.034
15	25	4	14779282	20009	1.75975	0.011
16	27	70	2.11E+08	939411	24.3368	0.248
17	29	41519	2.71E+09	26310531	304.8474	6.133
18	30	2238	9.15E+09	5936212	1131.248	1.71
19	32	113532	1.47E+11	2.88E+08	17779.31	84.69
20	33	506	5.28E+11	44061168	69967.83	13.847

References

[1] Dean S. Clark, "A Combinatorial Theorem on Circulant Matrices", *Amer. Math. Monthly*, Vol. 92, No. 10 (December 1985), pp. 725 ff. For those whose institutions are participants in JSTOR (Journal STORage), http://www.jstor.org/browse/#Mathematics provides access to this article — select American Mathematical Monthly, navigate to Vol. 92, No. 10, and search for "Clark". He later coauthored a paper on *n*-entry circular permutations. Dean S. Clark and Stanford S. Bonan, "Experimental Gambling System", *Mathematics Magazine*, Vol. 60, No. 4 (October 1987), pp. 217 ff. Again, http://www.jstor.org/browse/#Mathematics provides access — select Mathematics Magazine, navigate to Vol. 60, No. 4, then search for "Clark".

[2] Martin Gardner, "987654321", *Isaac Asimov's Science Fiction Magazine*, August 1986, p. 100.

[3] Timothy J. Rolfe, "Recurse Around the Clock", *Mathematics and Computer Education*, Vol. 21, No. 2 (Spring, 1987), pp. 98-104.

[4] See Problem E in http://www.acmcontest-pacnw.org/ProblemSet/2002/forweb.zip

[5] See http://icpc.baylor.edu/icpc/

[6] Timothy Rolfe, "Backtracking Algorithms", *Dr. Dobb's Journal*, Vol. 29, No. 5 (May 2004), pp. 48, 50-51.

[7] These implementations (which were developed as the instructor's solution when the problem was given as an assignment in an Algorithms course) are available through the following URL: http://penguin.ewu.edu/~trolfe/BoundClock/Implementation.html