Notes on Optimal Coalescing

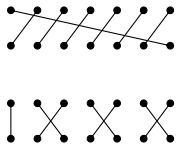
Daniel Grund[1]**

Department of Computer Science
Saarland University
grund@cs.uni-sb.de

Abstract. This short paper can be seen as an appendix to "A Fast Cutting-Plane Algorithm for Optimal Coalescing" [1] and that is exactly the way one should read it to grasp its full meaning.

1 Optimizing fixed points is only an approximation

This section deals with the subtle point that minimizing the number of instructions generated for a given permutation is *not* equivalent to maximizing its number of fixed points, but is far more difficult in fact. The smallest examples for this *anomalies* are shown in Figure 1.

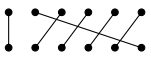


8 `moves`

9 `moves`

(a) Although the number of fixed points increased by one, the number of `move`s also increased.

3 `xchg`s



4 `xchg`s

(b) Similar anomaly applies for the number of `xchg`s.

Fig. 1. Smallest examples showing that the number of fixed points is not the only influencing factor concerning the number of `move`/`xchg` instructions.

Informally speaking, short and therefore more cycles are bad for copies, due to the saving copy per cycle. For exchanges in contrast, short cycles are better because two elements can be permuted by one transposition.

In general one has to distinguish between architectures with register exchange instructions and architectures only providing a copy instruction. Consider a permutation $\sigma \in S_n$ with f fixed points (cycles of length 1) and c non-trivial cycles. Remember, that for a cycle of length $m > 1$ one needs $m - 1$ transpositions or $m + 1$ copy instructions to implement it. Using copies is only possible if an

** Partially supported by the German Research Foundation (DFG) GK 623.

unused register is available. Otherwise one has to implement each transposition with 3 `xor`s. Finally, the number of instructions needed to implement such a permutation is given by:

$$\min \ \#\text{instructions} = \begin{cases} n-f-c & \texttt{xchg} \text{ instructions} \\ n-f+c & \texttt{mov} \text{ instructions} \\ 3(n-f-c) & \texttt{xor} \text{ instructions} \end{cases}$$

However, we are not aware of any "coalescing" algorithms attacking exactly these target functions. Instead coalescing algorithms up to now simply maximize the number of fixed points ($\max f$). But this only correlates with the above target functions, it does not necessarily optimize them. With an even number of registers r the difference between these two optimizations can be at most $r/2-2$ `mov` or $r/2-1$ `xchg` instructions per permutation.

However, in practice optimal solutions of $\max f$ produce a lot of fixed points and therefore limit the range for these anomalies quite well. The I386 architecture has only six or seven[1] usable registers. Thus, it is a good engineering decision to simplify the problem to only maximize the fixed points. For larger register files it may be necessary to attack the full problem. Without formal proof I consider it very unlikely to gain a measurable advantage.

2 The very details

2.1 Alternative ILP based Algorithms

- As described in [2]. Main difference: The decision variables that model the coalescing decision are "reversed" and other constraints have to be used in order to link them to the coloring variables.
- The constraints on our y-variables can be relaxed. For a correct model it is sufficient to restrict them to the real interval $[0,1]$. But the information that those variables are binary overweights the disadvantage of more non-continuous variables.
- The problem has a strong separation between the coloring and the coalescing variables, and only the coalescing variables occur in the objective function. It may be worthwhile to think about applying a Benders-decomposition.

2.2 **Tuning** CPLEX

The following settings turned out to be useful:

- set mip emph 2. New value for emphasis for MIP optimization: 2 (emphasize optimality)
- set mip ord 1. New value for type of generated priority order: 1 (decreasing cost)

[1] When using the frame pointer `ebp` is possible

2.3 Notes on Permutations

- Number of cycles in a permutation: min 1, avg H_n, max n, dev $\sqrt{H_n - H_n^{(2)}}$.
- Length of a random cycle: avg n/H_n.
- Number of singleton cycles: min 0, avg 1, max n, dev 1.

References

1. Grund, D., Hack, S.: A Fast Cutting-Plane Algorithm for Optimal Coalescing. In Krishnamurthi, S., Odersky, M., eds.: Compiler Construction - CC 2007. Lecture Notes In Computer Science, Springer (2007) Braga, Portugal.
2. Appel, A.W., George, L.: Optimal Spilling for CISC Machines with Few Registers. In: ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation. (2001) 243–253 This is the 2000-11 version.