Quadratic algorithm for cumulative edge-finding

Introduction

Edge-finding is a commonly used filtering algorithm for resource constrained scheduling problems. For cumulative scheduling problems (i.e., resources with a capacity greater than 1), the state of the art was until recently the Θ-tree algorithm by Vilím[3], with a complexity of $O(n \log n)$, where $n$ is the number of tasks and $k$ the number of distinct resource requirements.

We present a new edge-finding algorithm of $O(n^2)$ complexity, and demonstrate that in practice it outperforms earlier algorithms, while offering comparable performance to Vilím’s more recent timetable edge-finding [4].

Cumulative Resource Scheduling

In a resource constrained scheduling problem, a set of tasks share a finite capacity of a resource. Tasks have fixed resource requirements and durations, and a domain of start times. The problem is to assign each task a start time such that the capacity of the resource is never exceeded.

We present a new edge-finding algorithm of $O(n^2)$ complexity, with a complexity of $O(n \log n)$, with a speedup of about $O(n)$ factor on many instances that all other algorithms we tested timed out on.

Maximizing density intervals

To correctly locate $t$ when $est_0 > est_k$, we introduce the notion of maximum density, where density measures the average resource usage by a task interval.

Maximum density intervals

We prove that when $est_0 > est_k$, the interval $[t, t + \text{density}]$ with $est_0 > est_k$ with the maximum density is either (a) the interval $[t, t + \text{density}]$ that satisfies the inner maximization of $(EF)$, or (b) an interval where $est_0 > est_k$, which must also lead to a (possibly weaker) update of $est_0$. In the latter case, we further demonstrate that the algorithm moves closer to finding the true maximal update on each propagation, requiring in the worst case propagations on the order of $O(n)$; experimental results suggest that, in practice, our algorithm rarely requires more propagations than other edge-finders.

Results

The algorithm was implemented in Gecode 3.6.0, and compared with several previous edge-finding filters. The j30, j60 and j90 problem sets of the Project Scheduling Problem Library (PSPLib) were used as benchmarks. Runtimes are shown for the time required to find the best solution on a 3.07 ghz Intel Core 7 processor, with a time limit of 300 seconds.

Conclusion

- We present a $O(n^2)$ algorithm for cumulative edge-finding which does not suffer from the incompleteness of the original quadratic algorithm [1].
- Our algorithm outperforms the state of the art Θ-tree algorithm [3], while offering a simpler implementation with no custom data structures.

Acknowledgements: The authors would like to thank Christian Schulte, Pascal van Hentenryck and Petr Vilím for their suggestions; also the Swedish Research Council, for the third author is supported by grant 2009-4394.

Figure 1: 90m edge-finding[3] has a complexity of $O(n \log n)$, where $n$ is the number of distinct capacity requirements among the tasks, which is not strictly dominated by the $O(n^2)$ complexity of our algorithm, especially for small $k$. Nonetheless, in terms of runtime our algorithm consistently outperformed 90m, with a factor of three.

Figure 2: Despite the theoretical possibility that our algorithm might require additional propagation to check the same bounds as earlier algorithms (specifically for the Θ-tree algorithm[3]), here our algorithm nearly halves the number of propagations, and in almost all instances.

Figure 3: A comparison with Vilím’s $O(n \log n)$ timetable edge-finding[3] was not clear-cut. Each algorithm performed better on several instances, with timetable edge-finding performing better on difficult (j60) instances, but worse on easy (j30) and (j90) instances. Interestingly, the Vilím algorithm, which is strictly speaking stronger than traditional edge-finding, was able to solve only of the harder instances that all other algorithms were implemented on.

References