DEFERMENT CONTROL IN REOPTIMIZATION – HOW TO FIND FAIR REOPTIMIZED DISPATCHES

JÖRG RAMBAU

ABSTRACT. This note about work in progress suggests new policies for combinatorial online optimization problems where requests have to be served and the long-term objective is a sophisticated combination of request based cost (quality of service) and service based cost (operational cost). Examples are the online dispatching of automobile service units or the online control of cargo elevators. The new policies are reoptimization policies that do not use any (stochastic) information about future requests. For the first time, the new policies enhance reoptimization with a flow time guarantee, depending on the load in the system. In this sense, the new policies add fairness to reoptimization at a small cost: In the elevator control problem, e.g., the average performance w.r.t. the long-term objective function in simulation experiments only slightly worse than plain reoptimization for various long-term objectives.

1. INTRODUCTION

The German automobile association ADAC maintains a fleet of 1700 vehicles and has agreements with around 5000 service contractors. With these resources, they help people whose cars have broken down on the road. Those people can call an ADAC help center, and within 10 seconds, an assignment of a service resource to their request is made. At the same time, for all service vehicles, tours through the assigned requests have to be planned so as to minimize a certain (complicated) cost function for this so-called dispatch. No useful knowledge about future requests is available at the time being. Therefore, the current policy of the automated system, developed in joint work with Sven O. Krumke, is to reoptimize the whole dispatch upon the occurrence of each relevant event, like the arrival of a new request (see [2, 6, 5]).

A similar online-optimization problem appears in the pallet elevator group control in a large distribution center of Herlitz PBS AG in Falkensee near Berlin (see [1]).

The problem with reoptimization policies in general is that, depending on the reoptimization objective, an arbitrarily large deferment of individual requests can be observed (see [3] for an example). In a way, individual requests are sacrificed in favor of a good performance according to the reoptimization objective. Nevertheless, w.r.t. the reoptimization objective, the reoptimization policies in the long run usually perform much better than the currently known policies that can not cause infinite deferment. Therefore, the goal is to modify reoptimization policies so as to prevent deferment.

Sometimes deferment can be almost eliminated by enhancing the reoptimization objective with some terms that penalize waiting, but service in a fixed time can still

Dagstuhl Seminar Proceedings 05031
Algorithms for Optimization with Incomplete Information
http://drops.dagstuhl.de/opus/volltexte/2005/66

not be guaranteed, and this kind of objective function engineering is a very time consuming tuning issue, interfering with the original management objective.

It is known that under Δ-reasonable load, there is a policy called IGNORE that guarantees a maximal flow time of 2Δ for certain online optimization problems like the elevator control problem (see [4]). This policy, however, does not aim at optimizing any long-term objective function other than the maximal flow time. Therefore, in most practical applications, where it is desirable to minimize a certain cost function in the long run, IGNORE is outperformed by reoptimization policies that explicitly take the long-term cost function into account.

In this work, we combine the benefits of reoptimization with the best possible flow time guarantee, thereby solving the problem of infinite deferment for reoptimization policies. The crucial step is to add certain *flow time constraints* (which is no surprise) and *makespan constraints* (which might be less obvious) to the reoptimization problem. This yields the same worst-case bound on the maximal flow time as IGNORE in case the whole request set is reasonable (definitions see below). Moreover, the average performance w.r.t. the long-term objective in simulation experiments is almost as good as for the original reoptimization policies.

This construction can be done with *any* reoptimization problem, resulting in the same flow time guarantee. On the other hand, there is no theoretical guarantee for a good average performance w.r.t. the long-term objective, so our preliminary simulation results may not transfer to all other problems.

Note that, due to the nature of work in progress, the ideas of this notes have not yet been formalized in the most concise way.

2. PRELIMINARIES

For the sake of this note, we call any combinatorial online optimization problem where requests have to be served by a set of resources an *online service optimization problem*. The flow time of a request in a solution is the completion time of the request minus its enter time.

A *reoptimization policy* for an online service optimization problem, roughly speaking, bases its decision on a tentative plan. This plan contains information about how to proceed in case no new relevant external event would happen. The plan is updated whenever a relevant event occurs, e.g., the arrival of a new transportation request in the elevator example. The update is done according to optimal or near optimal solution of some reoptimization problem, an auxiliary offline optimization problem whose input data is taken from the current state of the system.

The cost function and the constraints of the reoptimization problem utilized by a reoptimization policy are called the *reoptimization* objective and constraints, resp. The cost function and the constraints of the original online service optimization problem are called the *long-term* objective and constraints, resp.

A *congruent* reoptimization policy uses the offline analogue of the online optimization problem as its reoptimization problem, i.e., the same constraints and the same objective function as the original problem. It has been observed that congruent reoptimization can lead to infinite deferment of individual requests.

In the following we assume that there is a fixed system state with no unserved requests, the *home state*. *Serving a set of request* means serving all requests in the sense of the original problem plus returning to the home state. The home state might be given by a canonical position of resources or the like.

The policy IGNORE uses a plan, like reoptimization policies. However, a new plan is only computed when the old plan is completely finished, i.e., the set of requests in the plan has been served. When the old plan is finished and there are still unserved requests, then a new plan is computed that minimizes the makespan, i.e., the time at which the plan is finished. Note that the work on a plan starts and ends in the home state of the system.

A request set is Δ-*reasonable* for an online service optimization problem and a home state for it if there is no time window T of width $\delta \geq \Delta$ such that the subset of requests released in T cannot be served in time δ. For Δ-reasonable request sets there is a policy called IGNORE that achieves a maximal flow time of 2Δ for all online service optimization problems.

3. RESULTS

Consider some online service optimization problem, and let $\Delta > 0$. A feasible solution of an instance of the congruent reoptimization problem satisfies the *flow time constraint w.r.t.* Δ if it schedules no request with flow time larger than 2Δ. Moreover, such a feasible solution satisfies the *makespan constraint w.r.t.* Δ if its makespan, i.e., the time at which the solution is finished and the system has returned to the home state, is no larger than Δ.

The reoptimization problem that is obtained from the congruent reoptimization problem by adding flow time and makespan constraints is called the *flow and makespan* Δ*-constrained reoptimization problem*. The reoptimization policy that updates its tentative plan by an optimal solution of the corresponding instance of the flow and makespan Δ-constrained reoptimization problem whenever this problem is feasible is called *flow and makespan* Δ*-constrained reoptimization policy* Δ-FMC-REPLAN. Whenever the instance of the flow and makespan Δ-constrained reoptimization problem is infeasible, the old plan is kept.

Theorem 1. *Under* Δ*-reasonable load, the policy* Δ-FMC-REPLAN *is a feasible policy that yields a maximal flow time of at most* 2Δ *for all service optimization problems.*

This is the best possible guarantee in the following sense: that there is a service optimization problem, namely the online dial-a-ride problem on a path graph, for which there is a Δ*-reasonable request set such that no policy can achieve a maximal flow time of strictly less than* 2Δ.

Sketch of the proof: In this short note, we can only provide the intuition behind the construction.

Intuitively, the Δ-FMC-REPLAN policy connects the features of congruent reoptimization and IGNORE: Assume, no reoptimization instance is feasible. Then Δ-FMC-REPLAN works the same as IGNORE, and this policy is known to have the asserted flow time guarantee. The flow time and makespan constraints w.r.t. Δ are constructed in such a way that any feasible solution accepted as a tentative plan will have all necessary properties that are used in the proof for the flow time guarantee of IGNORE (see [4]). In a sense, replacing the old plan by a feasible flow time and makespan Δ-constrained new plan does no harm to the flow time guarantee. □

There are some subtle details depending on the exact definitions that are not mentioned here. A more rigorous framework allowing for a formally precise statement and proof is still in progress.

A number of features make this approach quite attractive for practical online service problems.

- The policy Δ-FMC-REPLAN needs to know Δ. If Δ is unknown then a policy can be constructed that estimates Δ by the very common doubling technique (leading to overestimation and hence to weaker flowt time guarantees) or by makespan computations (leading to underestimation and hence to more restricted reoptimization problems).
- The flow time constraints w.r.t. Δ are the tightest possible constraints that guarantee feasibility of the policy under Δ-reasonable load. It is possible to relax these constraints arbitrarily in order to find a suitable trade-off between flow time guarantee and average performance w.r.t. the long-term objective.
- The theory can be extended to estimate the influence of utilizing suboptimal solutions when approximation algorithms are employed.

4. CONCLUSION

Whenever there is reason to believe that reoptimization policies may perform well for some online optimization problem, the new method of flow time and makespan constrained reoptimization will enhance them with a flow time guarantee that is best possible in the worst case under reasonable load.

How well this new policy performs on average w.r.t. the long-term objective is most probably problem dependent. This may foster a new line of research that evaluates the average performance of policies for specific online service optimization problems under reasonable load.

REFERENCES

[1] P. Friese and J. Rambau. Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models. ZIB-Report ZR 05-03, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2005. submitted to Discrete and Applied Mathematics, special issue on the Latin American Conference on Combinatorics, Graphs, and Applications (LACGA'04), Santiago, Chile, 2004.

[2] M. Grötschel, S. O. Krumke, J. Rambau, and L. M. Torres. Online-dispatching of automobile service units. In U. Leopold-Wildburger, F. Rendl, and G. Wäscher, editors, *Operations Research Proceedings*, pages 168–173. Springer, 2002.

[3] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. Preprint SC 99-08, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1999. Extended version of [4].

[4] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 137–149. Springer, 2000.

[5] B. Hiller, S. O. Krumke, and J. Rambau. Reoptimization gaps versus model errors in online-dispatching of service units for adac. *Electronic Notes in Discrete Mathematics*, 18:175–163, 2004.

[6] S. O. Krumke, J. Rambau, and L. M. Torres. Realtime-dispatching of guided and unguided automobile service units with soft time windows. In R. H. Möhring and R. Raman, editors, *Algorithms – ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17–21, 2002, Proceedings*, volume 2461 of *Lecture Notes in Computer Science*. Springer, 2002.

LEHRSTUHL WIRTSCHAFTSMATHEMATIK, UNIVERSITÄT BAYREUTH, 95440 BAYREUTH
E-mail address: joerg.rambau@uni-bayreuth.de