Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

BENJAMIN HILLER SVEN O. KRUMKE JÖRG RAMBAU

Reoptimization Gaps versus Model Errors in Online-Dispatching of Service Units for ADAC

ZIB-Report 04-17 (April 2004)

REOPTIMIZATION GAPS VERSUS MODEL ERRORS IN ONLINE-DISPATCHING OF SERVICE UNITS FOR ADAC

BENJAMIN HILLER, SVEN O. KRUMKE, AND JÖRG RAMBAU

Abstract. Under high load, the automated dispatching of service vehicles for the German Automobile Association (ADAC) must reoptimize a dispatch for 100–150 vehicles and 400 requests in about ten seconds to near optimality. In the presence of service contractors, this can be achieved by the column generation algorithm ZIBDIP. In metropolitan areas, however, service contractors cannot be dispatched automatically because they may decline. The problem: a model without contractors yields larger optimality gaps within ten seconds. One way-out are simplified reoptimization models. These compute a short-term dispatch containing only some of the requests: unknown future requests will influence future service anyway. The simpler the models the better the gaps, but also the larger the model error. What is more significant: reoptimization gap or reoptimization model error? We answer this question in simulations on real-world ADAC data: only the new model ZIBDIP$_{\text{dummy}}$ can keep up with ZIBDIP.

1. Issues and Motivation

Currently, the German Automobile Association (ADAC) evaluates an automated dispatching system for service vehicles (units) and service contractors (conts) on the basis of exact cost-reoptimization. This means that a current dispatch is maintained, which contains all known yet unserved requests and which is near optimal on the basis of the current data; whenever a unit becomes idle its next request is read from the current dispatch; at each event (new request, finished service, etc.) the dispatch is updated by a reoptimization run.

A feasible current dispatch for all known requests and available service vehicles is a partition of the requests into tours for units and conts such that each request is in exactly one tour and each unit drives exactly one tour (maybe directly to its home position) so that the cost function is minimized. Cost contributions come from driving cost for units, fixed service costs per requests for conts, and a strictly convex lateness costs for violation of soft time windows at each request (currently quadratic). The latter cost structure is chosen so as to avoid large individual waiting times for customers.

It is not a-priori clear that such a rigorous reoptimization yields the best, or even a good, long-term cost (this is the *online issue* of the dispatching problem). Indeed, at times in the literature it is claimed that exact reoptimization (i.e., with small optimality gap) does not pay in practice because of the unknown future requests [1, p. 5]. In the case of this particular application, however, the results of exact reoptimization are satisfying [2], in concordance with [3, Sec. 8.4].

Although the reoptimization problem, which is modeled as a set partitioning problem for tours, has an astronomical number of variables, it can be solved by a dynamic column generation procedure. An effective method to obtain provably good

Key words and phrases. vehicle dispatching, real-time, integer linear programming, dynamic column generation, dummy contractor, shadow price model.

Supported by the DFG research center "Mathematics for key technologies" (FZT 86) in Berlin.".

solutions in ten seconds (this is the *real-time aspect* of the dispatching problem) is *dynamic pricing control*, which is the main feature of our ZIBDIP algorithm [4].

As it turns out, the fixed cost for service by conts bound the dual values of requests. Thus, conts substantially contribute to the success of the branch-and-bound method underlying the dynamic pricing control in ZIBDIP. In metropolitan areas, however, there is a problem with the automated dispatching of conts: the cont may decline to serve suggested requests, in which case this request has to be reentered into the system. Since the majority of the conts is not connected to the computer network of ADAC each assignment of a request to a contractor involves a manual phone call. Because conts delined too often, the dispatching process at ADAC was slowed down and, as a consequence, the ADAC decided to remove conts from the model.

In simulations on ADAC production data (three days in December 2002 with high load) without conts, we encountered a significant reoptimization gap. For 2002/12/13, e.g., Fig. 1 shows the gap of the reoptimization result to the respective lower bound coming from the optimal solution of the LP relaxation (this lower bound was computed a-posteriori for each reoptimization). The reoptimization still works well in most cases, but under high load the solutions – delivered after ten seconds – exhibit optimality gaps around 3% on average but up to 15% in peak load situations.

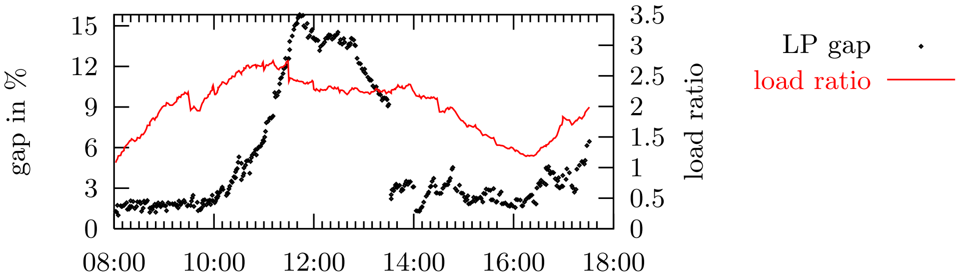


FIGURE 1. Integrality gap over time of ZIBDIP (the load ratio is the number of requests per unit in a reoptimization problem)

One way to overcome this problem is to consider simplified reoptimization models that stem from the following considerations: In principle, for each unit we only have to determine the next request to work on. The complete dispatch is computed only to pick up future synergies by considering more than one request per unit. Synergies that are implemented only very far in the future will be disturbed by new requests anyway; therefore, an exact pre-calculation of the best decisions in, say, two hours may not really be necessary; consequently, one can try to consider only a subset of requests in a reoptimization step.

The issue of this experimental work is: should one stick to the complete model and accept occasional substantial reoptimization gaps or is it better to simplify the reoptimization model so as to eliminate the reoptimization gap? This question is answered on the basis of simulation studies, performed on the aforementioned ADAC production data: we first compare the original ZIBDIP reoptimization to several methods to select subsets of requests for reoptimization. Then ZIBDIP competes with a simple online heuristic for the ZIBDIP model in order to estimate how even larger reoptimization gaps harm in the long run.

2. MAIN RESULTS

We developed and evaluated the following strategies for request selection:

4-ZIBDIP: Select those requests that are among the four closest to some unit. This can be generalized to k-ZIBDIP.

PTC (Prescribed Total Cover): Relax the set partitioning condition to set packing, and require that a request set of cardinality twice the number of units is covered by tours of units. This requires at least that many requests in the system, which is the case under high load; under low load this may be infeasible, so one has to switch accordingly.

ShadowPrice: Solve the LP relaxation of ZIBDIP. To find an integral solution, relax the set partitioning condition to set packing and change the cost of each tour to its reduced cost from the hopefully near optimal LP solution. In this model, requests are assigned to units only if their LP dual prices pay enough to weigh out the primal cost of their service. This requires that the LP relaxation can be solved fast.

ZIBDIP$_{\mathbf{dummy}}$: Introduce a dummy contractor. This contractor can be assigned arbitrarily many requests at the same time, i.e., in reality, these requests are unassigned for the moment. The arrival time of the dummy contractor at any request is a fixed time, the dummy contractor delay. In our case, 135 minutes were chosen. This implies that, in an optimal solution, for any request in a tour of a unit, service will start after at most 135 minutes after reoptimization; otherwise, the request would have been assigned to the dummy contractor.

We furthermore evaluated a reoptimization heuristic for the original model:

BestInsert: Reoptimization is done by taking the dispatch of the previous reoptimization, removing all requests that have been served in the meantime, and inserting new requests at minimal additional cost w.r.t. to the original ZIBDIP-model.

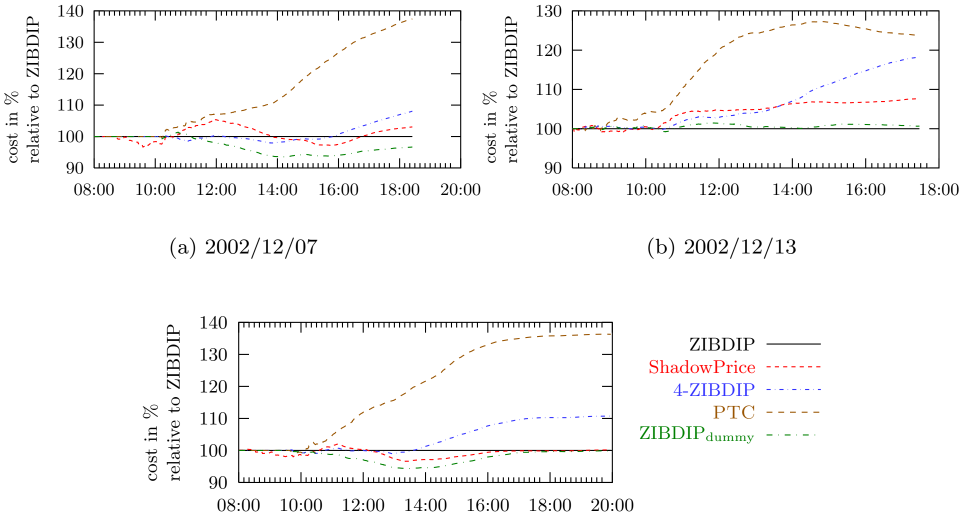
One should mention that in each reoptimization with either model, the solutions of the previous reoptimization are reused as start solutions – a simple but essential technique to stabilize the dispatching process in case of occasional suboptimal reoptimization.

instance	**requests**	**units**	**requests per unit**
2002/12/07	2123	125	16.98
2002/12/13	2537	146	17.38
2002/12/14	1731	131	13.21

Table 1. Sizes of high load instances used for simulation

The simulation data stems from three days of production at ADAC in December 2002; instance sizes are given in table 1. Depending on the instance, between 1700 and 2100 reoptimization runs where triggered. The software ran on a standard Linux PC, 2.4GHz Pentium 4 CPU, 4GB RAM, distribution Suse 9.0, kernel 2.4.21-202-smp, LP solver CPLEX 8.0, compiled with gcc 3.3.1. Each reoptimization run was interrupted after 10s run-time.

The results: only ZIBDIP$_{\text{dummy}}$ is competitive against ZIBDIP. In two out of three instances it has even slightly lower long-term cost than ZIBDIP, though by a small margin. Yet, the answer to our question is that the model error of most of our high-load models is worse than the computational error that ZIBDIP produces (Fig. 2). Therefore, such model reductions have to be treated with great care. In our case, ZIBDIP$_{\text{dummy}}$ seems to deliver the overall best solution. One needs to be careful, though: a substantially smaller contractor delay of 45 minutes would remove the reoptimization gap completely, it, however, would at the same time

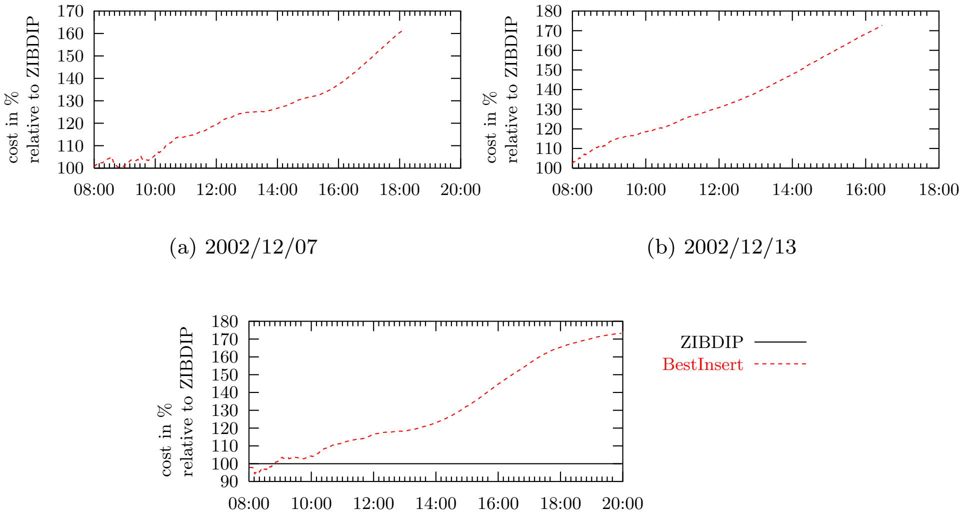


(a) 2002/12/07

(b) 2002/12/13

(c) 2002/12/14

Figure 2. ZIBDIP vs. simplified models



(a) 2002/12/07

(b) 2002/12/13

(c) 2002/12/14

Figure 3. ZIBDIP vs. BestInsert

produce unacceptable long-term costs because too many requests stay unassigned for too long.

That larger computational errors in the reoptimization can nevertheless spoil the long-term cost more significantly than the model errors above, is shown by the bad performance of the simple BestInsert heuristic for the original reoptimization model of ZIBDIP (Fig. 3). Thus, also in the dynamic setting it pays off to optimize as exactly as possible.

3. Significance

The production software for automated dispatching of ADAC service vehicles is produced by Intergraph Public Saftety (IPS), based on the ZIBDIP algorithm. In the view of the results presented in this work, ADAC has filed a change request for the production software: $\mathrm{ZIBDIP}_{\mathrm{dummy}}$ is now the standard reoptimization model because it has proven to be more robust against sudden load increase. The key learning is that rigorous reoptimization on the basis of mathematical programming – though myopic w.r.t. unknown future requests – yields the best results in this particular application.

References

[1] K. Q. Zhu, K.-L. Ong, A reactive method for real time dynamic vehicle routing problem, in: Proceedings of the 12th ICTAI, 2000.

[2] M. Grötschel, S. O. Krumke, J. Rambau, L. M. Torres, Online-dispatching of automobile service units, in: U. Leopold-Wildburger, F. Rendl, G. Wäscher (Eds.), Operations Research Proceedings, Springer, 2002, pp. 168–173.
URL `http://www.zib.de/PaperWeb/abstracts/ZR-02-44/`

[3] D. Bertsimas, D. Simchi-Levi, A new generation of vehicle routing research: robust algorithms, addressing uncertainty, Operations Research 44.

[4] S. O. Krumke, J. Rambau, L. M. Torres, Realtime-dispatching of guided and unguided automobile service units with soft time windows, in: R. H. Möhring, R. Raman (Eds.), Algorithms – ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17–21, 2002, Proceedings, Vol. 2461 of Lecture Notes in Computer Science, Springer, 2002.
URL `http://www.zib.de/PaperWeb/abstracts/ZR-01-22`

E-mail address: `krumke@mathematik.uni-kl.de`
E-mail address: `{hiller,rambau}@zib.de`

{Benjamin Hiller, Jörg Rambau}, Department Optimization, Zuse-Institute Berlin, Germany

Sven O. Krumke, Department of Mathematics, University of Kaiserslautern, Germany