配送計画への制約ソルバーの適用

Solving Dispatching Problems with Constraint-based meta-heuristics

03000500 ｱｲﾛｸﾞ(株) ｶﾀｲ ﾌｪﾚﾝﾂ – ILOG Japan Co.、Katai Ferenc

1. Introduction

Dispatching problems are generally considered difficult to solve, due to their combinatorial nature. It is especially true for industrial problems, where many side constraints exist, such as vehicle breaks, parking-lot capacity, possibility to split or group orders, etc. In the literature there are several attempt to use evolutionary algorithm, such as GA (Potvin et al[3], simulated annealing (Thangiah et al[5]), tabu search (Taillard et al[4], deBacker et al[1]), etc and hybrid approaches (Chiang et al[2], Thangiah et al[5]). In the present paper we introduce a 2-phase approach, using a combination of constraint programming (CP) and (meta)-heuristics (local/tabu searche). The algorithms and framework has been implemented in ILOG Dispatcher C++ library (an add-on of ILOG Solver (CP) library) together with a modeling layer called ILOG Concert technology. Though, we focus on benchmark examples, industrial scale applications implemented with ILOG Dispatcher are numerous in the domain of truck, technician dispatching, pick-up and/or delivery problems, etc. either in standalone fashion, or part of ISVs' (independent software vendor) packages.

2. The Dispatcher model – solving method

A solution for a dispatching problem means, of each truck is decided if it is used, and visits are assigned to used trucks in appropriate order (respecting various constraints). The method comprises of 2 steps. A first solution (by any standard) constructed using different heuristics by choice – in Figure 1 the result of the so-called sweep heuristics can be seen.

After generating the first solution, in the next phase the solution is improved by either (improve) heuristics (move is accepted, when it strictly reduces the cost) or meta-heuristics, such as tabu-search (TS) and guided local search (GLS) (solutions can be degraded, that is, move can be accepted even if it doesn't reduce the cost)). In fact, providing move operators, such as 2opt, or-opt, exchange, etc, the user can build his own custom-made heuristics, or meta-heuristics. During the moves, the CP engine is propagating the changes to verify if the move is feasible. If the move is not feasible the result of the move is discarded and other moves are explored. Heuristics can't provide the proof of optimality, so the stopping criteria either limiting searching time and/or the number of accepted moves.

3. Problems and constraints

ILOG Dispatcher has many built-in constraints, such as expressing: first and/or last visit to depot(s) or customer site(s) by a vehicle; truck's capacity (compartments-dimensions), speed and time window constraint of availability; time windows of depot use; alternative sites and time windows for a visit; vehicle break (effectively driver's break); max and min running time of vehicle; on-route pick-up and delivery; multiply depots; loading/unloading fixed and unit time; vehicle speed, compatibility constraints; precedence of visits etc. Besides, based on the underlying CP engine and the provided variable objects, such as: transition, travel and cumulative variables between visits for time, distance and capacity; waiting time and delay variable; variable of next and previous visits etc. many custom constraints could be expressed, such as: allowing-disallowing splitting orders (10kg banana + 10l milk is delivered by the same truck or by different trucks); area related constraints; parking-lot capacity etc.

4. Bechmark results

Tests were conducted on several different benchmark problems of Solomon[4]. Some of the results by guidedTS (GTS) and fastGLC (fGLS) can be seen in Table 1. These problems have in general, 100 visits, a central depot, capacity constraints, time windows on the time of delivery, and a total route time constraint. The objective was to minimize traveled distance, meanwhile keeping the number of trucks as low as possible. A site has x, y coordinates, and the distance is computed by Euclidean distance. After generating the first solution, the improve-heuristics was used up to the local minimum. Then GTS and fGLS were used for meta-heuristics. The results were obtained with a 300s time limit and 3000 accepted moves, on a fairly slow computer (200MHz Pentium (bus-speed 66Mhz)). Δ gives the ratio of our result and the best one (found so far in the literature) in %. An asterisk means that a solution with the minimum number of vehicles was not found during the time elapsed. In Figure 2 and 3 the shape of the routes of C101 and C201 can be seen, respectively.

5. Conclusion

One could exploit more specific data or constraints of the problem at hand to design problem-specific algorithms for dispatching problems. However, the 2-phase, 2-technology approach proved to be robust and fast. Meanwhile, there is plenty of room for improvement, the quality of solutions is more than promising when the user wants to turn to industry size problems. Reasoning on the constraints, that is, improving the underlying CP engine is necessary. Also a key demand is to enable the use of capacity resources at the site (warehouse) of a visit, enabling safety-level, max capacity, etc. constraints.

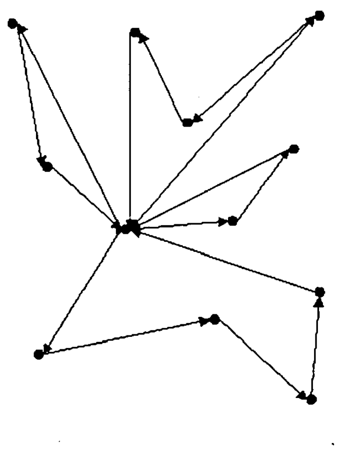

Fig. 1 First solution with sweep heuristics

Problem	GTS	Δ(%)	fGLS	Δ(%)
C101	828.94	0.00	828.94	0.00
C102	832.48	0.31	829.26	0.04
C103	828.99	0.00	828.07	0.00
C104	825.87	0.00	824.78	0.00
C105	828.94	0.00	828.94	0.00
C106	828.94	0.00	828.94	0.00
C107	828.94	0.00	*	*
C108	828.94	0.00	828.94	0.00
C109	828.94	0.00	828.94	0.00
C201	591.56	0.00	591.56	0.00
C202	591.56	0.00	591.56	0.00
C203	591.56	0.06	609.12	3.04
C204	599.76	1.55	720.58	20.79
C205	588.88	0.00	588.88	0.00
C206	588.88	0.07	588.49	0.00
C207	588.88	0.10	588.27	0.00
C208	588.88	0.09	588.32	0.00

Table 1. Results for some Solomon problems

At the moment though it is possible, using the *reservoir* resource object of ILOG Scheduler (another add-on to ILOG Solver for solving scheduling problems), it is desirable to move this concept into the dispatching framework reducing the necessity to duplicate modeling objects, consequently reducing memory usage and increase performance and to ease modeling

References

[1] B. de Backer, V. Furnon, P. Kilby, P. Prosser, and P. Shaw: "Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics", *Journal of Heuristics*, Volume No., 1–21 (1997)

[2] W.C. Chiang and R.A. Russell: "Hybrid heuristics for the vehicle routing problem with time windows", *Department of Quantitative Methods, University of Tulsa, Tulsa, OK* (1993)

[3] J.-Y. Potvin, S. Bengio: "The vehicle routing problem with time windows - part ii: Genetic search", *INFORMS Journal on Computing*, 8:165–172 (1996)

[4] M. M. Solomon: "Algorithms for the vehicle routing and scheduling problem with time window constraints", *Operations Research*, 35:254–265 (1987)

[4] E. Taillard, P. Badeau, M. Gendreau, F. Guertain, and J.-Y. Potvin: "A tabu search heuristic for the vehicle routing problem with soft time windows", *Transportation Science*, 32(2) (1997)

[5] S. R. Thangiah, I. H. Osman, and T. Sun: "Hybrid genetic algorithm, simulated annealing, and tabu search methods for vehicle routing problems with time windows", *Working paper UKC/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury* (1994)

[6] ILOG Dispatcher3.0, User's Manual, Copyright © 2000 ILOG, http://www.ilog.com

[7] ILOG Solver5.0, User's Manual, , Copyright © 2000 ILOG, http://www.ilog.com

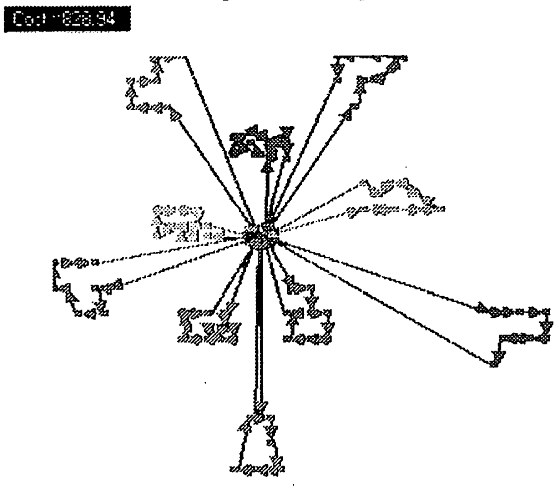

Fig. 2 Solution of C101 with cost 828.94

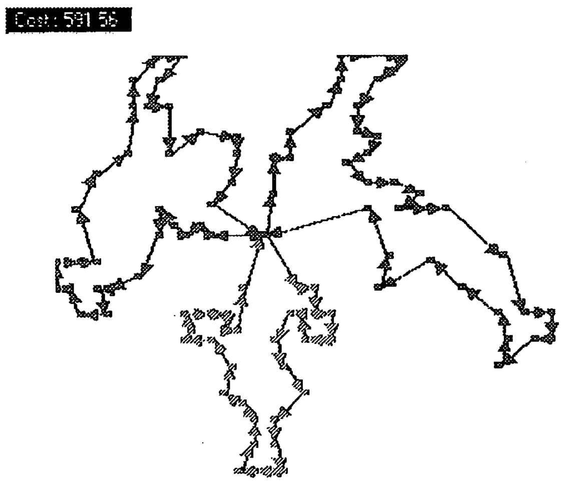

Fig. 3 Solution of C201 with cost 591.56