A Parallel Algorithm for Minimizing the Fleet Size in the Pickup and Delivery Problem with Time Windows

Miroslaw Blocho
ABB IT
Zeganska 1
04-713 Warsaw, Poland
miroslaw.blocho@pl.abb.com

Jakub Nalepa
Silesian University of Technology
Akademicka 16
44-100 Gliwice, Poland
jakub.nalepa@polsl.pl

ABSTRACT

In this paper, we propose a parallel guided ejection search algorithm to minimize the fleet size in the NP-hard pickup and delivery problem with time windows. The parallel processes co-operate periodically to enhance the quality of results and to accelerate the convergence of computations. The experimental study shows that the parallel algorithm retrieves very high-quality results. Finally, we report 13 (22% of all considered benchmark tests) new world's best solutions.

CCS Concepts

•**Computing methodologies** → **Parallel algorithms; Planning and scheduling;**

Keywords

PDPTW; MPI; heuristics

1. INTRODUCTION

Tackling complex transportation problems is a core issue in logistics and planning. Solving the pickup and delivery problem with time windows (PDPTW) consists in determining the plan to serve requests using a fleet of vehicles. Each request encompasses delivering goods from one location to another. All vehicles start and finish at the depot, the amount of goods must not exceed the vehicle capacity, each customer must be served within its time window, and the pickup is handled before the corresponding delivery [2].

Exact algorithms for the PDPTW are still not applicable to large-scale scenarios, and approximate algorithms (tabu searches, simulated annealing, guided ejection searches, and numerous other) became a main stream of research [4]. Parallel heuristics deliver high-quality solutions to a wide variety of problems quickly. The co-operation of processes in these algorithms is crucial since it helps guide the search [3].

We propose a parallel guided search (P–GES) to minimize the fleet size in the PDPTW. It extends the guided search (GES) [2]. The Message Passing Interface (MPI) implementation is examined on Li and Lim's tests—experiments amount to nearly 40,000 CPU hours on an SMP cluster. We analyze its speedup and efficiency, and report 13 new world's best solutions obtained using P–GES.

EuroMPI '15 September 21-23, 2015, Bordeaux , France

ACM ISBN 978-1-4503-3795-3/15/09.

DOI: http://dx.doi.org/10.1145/2802658.2802673

2. PARALLEL ALGORITHM

P–GES contains p co-operating components. First, an initial solution σ is constructed, in which every request is served by a single truck. Attempts to reduce the number of routes (K) are then carried out until the computation time exceeds τ_M—a random route r is removed, and its requests are put into the ejection pool (EP). The penalty counters $\mathcal{P}$ for all requests are reset. Then, a request h_{in} is taken from the EP to be put back into σ. If there are more feasible insertions of h_{in}, a random one is drawn. Otherwise, an infeasible solution is squeezed—a random infeasible route r is affected by local moves (out-relocate and out-exchange) to decrease its penalty. σ' with the minimum penalty replaces σ (if its penalty is smaller). If the squeeze fails, $\mathcal{P}[h_{in}]$ is increased, and other requests are ejected. The solution with the minimum sum of the counters of ejected requests replaces σ, and is finally perturbed by out-relocates and out-exchanges.

The processes co-operate every τ_{coop} sec. Each co-operation is started by P_1, which sends its best σ to the next process in the ring (P_2) (Fig. 1). It compares the received solution with its best one, and sends the better one (with a lower K) to the next P, and so forth. Processes asynchronously wait for completion of started operations. We use a two-step data passing: first, only K is sent, along with the information whether σ will be sent next (i.e., if K has changed).

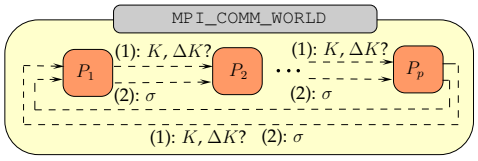

Figure 1: Communication scheme in MPI.

3. EXPERIMENTAL RESULTS

P–GES was implemented in C++ and run on the SMP cluster, on Intel Xeon Quad Core 2.33 GHz nodes (12 MB level 3 cache), connected by the Infiniband DDR fat-free network (20 Gbps, del. 5 μs). It was tested on 800-customer

PDPTW tests[1]. τ_M varied between 45 min. ($p > 8$), 60 min. ($p = 8$), and $p = 240$ min. ($p = 1$), and $\tau_{coop} = 5$ sec. P–GES was run 3× for each instance and p. We introduce the notion of a *target solution*. The arbitrary K_T (as minimum as possible) for a given test was used as the target. The speedup (S) was calculated as the ratio between the average time of the sequential algorithm ($p = 1$), and the average P–GES time ($p > 1$), necessary to converge to K_T routes.

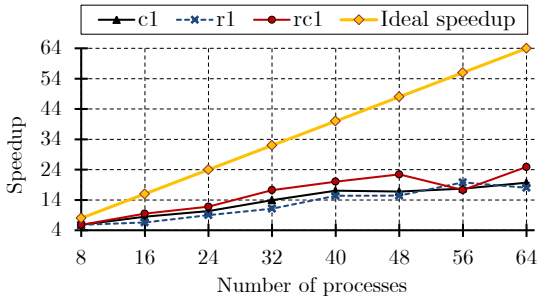

Figure 2: Speedup of P–GES (c1, r1, rc1 tests).

The S values are illustrated in Figs 2–3. The best efficiency (E) was achieved for $p = 8$ (0.6315) and $p = 16$ (0.5024), while $\bar{E} = 0.4181$. The best S was obtained for mixed customers (clustered and random on the map) with wide time windows served by large-capacity trucks (rc2). The worst S was elaborated for random customers (r1 and r2)—these tests are easier to solve even using the sequential algorithm. However, note that decreasing K beyond K_T is most challenging. We experienced the superlinear performance (in which $S > p$). There are 3 possible sources of this phenomenon [1]: (i) the implementation issues, (ii) numerical sources (e.g., the order of traversing the search space), and (iii) physical sources (e.g., cache effects). Although (ii) and (iii) seem to be likely sources of the superlinearity, it requires investigation. Interestingly, in the majority of rc2 instances (7 out of 10), executing P–GES resulted in $S > p$.

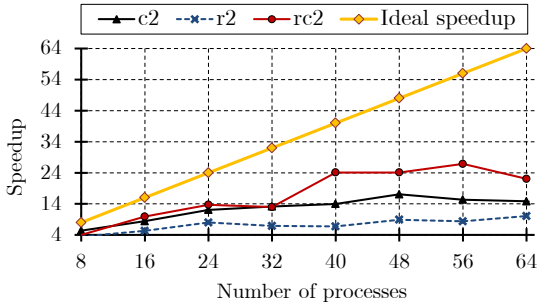

Figure 3: Speedup of P–GES (c2, r2, rc2 tests).

Increasing p shortens the convergence time (Fig. 4). For some tests, the communication overhead and possibly the work imbalance cause its increase ($p \geq 40$). The co-operation should be thus dynamic to avoid too frequent data exchange.

Tab. 1 shows the cumulated K—the number of vehicles in the best solutions for all instances in each class (WB

[1]http://www.sintef.no/projectweb/top/pdptw/li--lim-benchmark/800-customers/ (May 2, 2015).

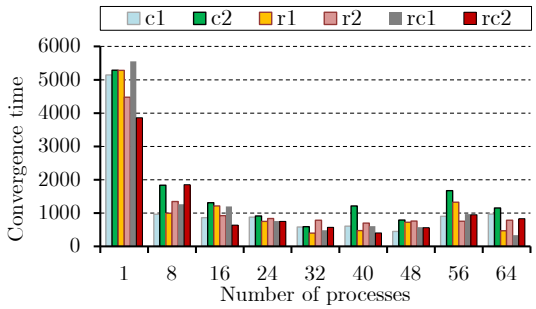

Figure 4: Convergence time (in sec.) of P–GES.

are the world's best K's). P–GES run on a larger p retrieved the best solutions, and 13 new world's best solutions (see: http://sun.aei.polsl.pl/~jnalepa/EuroMPI2015) were found. To verify the null hypothesis: "P–GES run on various p's leads to similar solutions", we performed two-tailed Wilcoxon tests. P–GES with $p \geq 56$ gives notably different results than run on lower p's (p-value is < 0.05).

Table 1: Cumulated K (best results are boldfaced).

$p \downarrow$	c1	c2	r1	r2	rc1	rc2	Sum
1	764	266	436	**109**	514	159	2248
16	761	266	**432**	110	509	152	2230
32	759	269	435	113	509	151	2236
48	759	265	433	110	**507**	152	2226
64	**753**	**260**	433	111	508	**149**	**2214**
WB	745	241	427	98	513	150	2174

4. CONCLUSIONS

We presented a parallel ejection search (P–GES) for minimizing K in the PDPTW. The experimental study gave detailed insights into its performance. We experienced the superlinear performance for several test cases. Finally, we reported 13 new world's best solutions found using P–GES.

Acknowledgments

This research was supported by the National Science Centre under research Grant No. DEC-2013/09/N/ST6/03461. We thank the Academic Computer Centre in Gdańsk (TASK CI), where the computations of our project were carried out.

5. REFERENCES

[1] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82(1):7 – 13, 2002.

[2] Y. Nagata and S. Kobayashi. Guided ejection search for the pickup and delivery problem with time windows. In *Proceedings of EvoCOP*, volume 6022 of *LNCS*, pages 202–213. Springer, 2010.

[3] J. Nalepa and M. Blocho. Co-operation in the parallel memetic algorithm. *International Journal of Parallel Programming*, pages 1–28, 2014.

[4] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.