Small-Moves Based Mutation For Pick-Up And Delivery Problem

Viacheslav Shalamov
ITMO University
49 Kronverksky av.
St. Petersburg, Russia
shalamov@rain.ifmo.ru

Andrey Filchenkov
ITMO University
49 Kronverksky av.
St. Petersburg, Russia
afilchenkov@niuitmo.ru

Daniil Chivilikhin
ITMO University
49 Kronverksky av.
St. Petersburg, Russia
chivdan@rain.ifmo.ru

ABSTRACT

One of the common approaches to solve the Pickup and Delivery problem is to use small local changes in already built solution called "small moves heuristics". In this paper we consider four different variants of these small local changes and one combination of them. Each of these strategies we used as a mutation operator in five different evolutionary algorithms. Experimental results indicate that applying the combined small moves strategy together with the $K + KN$ evolutionary algorithm yields the best results.

1. INTRODUCTION

Logistic and shipping management produces a family of routing problems, which have been a focus of the optimization community research. The most known problem is the Travelling Salesman Problem (TSP), which is often considered to be one of the main benchmarks for combinatorial optimization algorithms [8]. A variety of additional conditions and constraints exist for TSP. They create a huge family of optimization problems being NP-hard. If the travelling salesman is required not only to visit each point in his route, but also has to pick items in some points and deliver them to other points, the TSP with pickup and delivery (TSPPD) arises [7]. This problem belongs to a wider class of Pickup and Delivery Problems (PDP), where not only a single moving object (travelling salesman), but a set of moving objects (vehicles) is available to perform picking up and delivering.

The PDP class consists of many different routing problem variants, a recent review can be found in [10]. One-to-one PDP subclass includes only such PDPs, in which every vertex has only a single item, which should be delivered to another vertex [2]. Problems of this subclass, in which only a single vehicle is used, are called Single Vehicle PDPs (SVPDPs). Among them, a special type of problems is distinguished called SVPDP with LIFO (SVPDPL): another restriction should be satisfied that the vehicle stores items according to the last-in-first-out principle [4].

The SVPDPL problem arises in transportation of heavy, fragile or dangerous items, which should be taken out of a vehicle in the reverse order to the one they were put in [5]. Another aspect of the problem importance is its theoretical simplicity, which is useful for the general approaches and technique analysis.

SVPDPL is an NP-hard problem, thus, a number of heuristics for solving are suggested. Variable neighborhood search (VNS) presented in [1] is a search framework for applying local changes to a route in order to obtain a new, better one. These changes can be considered as mutation operators in terms of evolutionary computation [11], or edges in solution graphs [3]. Several change operators have been suggested or adopted in order to be applied with VNS. To the best of our knowledge, these operators have never been compared to each other, and even the problem of selecting these operators has never been stated.

In this paper, we undertake a comparison of five local change operators by conducting computational experiments on artificial data. The results of experiments are presented and discussed.

The remainder of the paper is organized as follows. In Section 2, we define SVPDPL, its solution and local changes, which can be applied with VNS. In Section 3, we describe algorithms we used for comparison and experiments for their performance evaluation, and in Section 4 we conclude the paper and outline the future work.

2. PROBLEM STATEMENT AND SMALL MOVES

2.1 SVPDP

First, we introduce SVPDP. We will partly follow the notation from [5]. Consider a weighted graph $G = (V, E, W)$, where V is the vertex set, E is the edge set, and W is a weight function defined on V and E, which will be discussed later. On the vertex set, a set of pairs is defined: $(P, D) = ((p_1, d_1), \ldots, (p_L, d_K))$, $P, D \in V$; P is a set of sources, where items should be picked up, and D is a set of corresponding destinations, to which the items should be delivered. We will enumerate vertices in the first subset with $1, \ldots, L$, and vertices in the second subset with $L+1, \ldots, 2L$.

Weight assigned to an edge represents the cost of travelling on this edge, and weight assigned to a vertex represents the load for the vehicle. In the simplest, case the edge weight is the Euclidean distance, and all the loads are $+1$ and -1 for pickup and delivery correspondingly.

To state SVPDPL as an optimization problem, we need to introduce more notations. Let x_{ij} be binary variables,

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA

DOI: http://dx.doi.org/10.1145/2908961.2931666

such that $x_{ij} = 1$ iff edge (i, j) belongs to the current route (PDP solution). For a vertex subset S, let $\delta(S)$ denote a set of edges, which have exactly one vertex belonging to S. For $E' \subseteq E$ let $x(E') = \sum_{P(i,j)\in E'} x_{ij}$. And for $S \subseteq V$ let $x(S) = \sum_{i,j\in S} x_{ij}$. Under these notations, the SVPDPL problem is to find a Hamilotonian cycle (path visiting each vertex only once and returning to the origin node) in the graph, which satisfies the following requirements:

1. Minimize $\sum_{i=0}^{2n} \sum_{j=i+1}^{2n+1} c_{ij} \cdot x_{ij}$.
2. $x_{0,2n+1} = 1$.
3. $\forall i \in V$, $x(\delta(i)) = 2$.
4. $\forall (i, j) \in E$, $x_{ij} = 0$ or $x_{ij} = 1$.

2.2 Small moves

Since the search of the exact solution is NP-hard, heuristic known as "small moves" was suggested [1, 11]. The main idea is to apply small changes to a found solution in order to obtain a new, better one.

In this work, we use the following four types of small moves.

Lin-2-Opt substitutes two edges (i_j, i_{j+1}) and (i_k, i_{k+1}) with other two edges (i_j, i_k) and (i_{j+1}, i_{k+1}), then it inverses the subpath $(i_{j+1}, \ldots, i_k)$.

Double-bridge, which was first introduced in [6], exchanges edges (i_j, i_{j+1}), (i_k, i_{k+1}), (i_l, i_{l+1}) and (i_h, i_{h+1}) with (i_j, i_l+1), (i_k, i_{h+1}), (i_l, i_{j+1}) and (i_h, i_{k+1}).

Couple-exchange selects two orders $(i, n+i)$ and $(j, n+j)$, then it replaces i with j and $n + i$ with $n + j$.

Point-exchange is analogous to the couple-exchange. The only difference is that not the pairs, but single points are exchanged.

We also use the "Mixed" small move where each of the four small moves can be applied with equal probability.

3. ALGORITHM AND EXPERIMENTS

3.1 Algorithms

The small moves, described in the previous sections, are usually applied in the following manner: we randomly pick a possible small move, and if its application improves the solution, we retain it. However, it is obvious that such a greedy-like strategy misses many possible solutions.

One can notices that this small-move technique is very close to mutation operations. The described strategy of application is just a single way to schedule different small moves. We suppose that various strategies from evolutionary computations can outperform simple strategy. We used five schemes based on simple evolutionary algorithms (EAs) — (1+1)-EA, (1+λ)-EA, (1, λ)-EA, and (μ + λ)-EA:

1. 1 + 1: each generation consists of a single individual which gives birth to a single child.
2. 1 + N: each generation consists of a single individual which gives birth to $N = \sqrt{L/2}$ children. In this work we used N equal to 5.
3. 1, N: each generation consists of a single individual which gives birth to N children and cannot be selected to the next generation.
4. 1 + N + Big mutation (1+N+BM): this is similar to 1+N, but generations with indices $3 \cdot L^2$, $6 \cdot L^3$, and $9 \cdot L^2$ are influenced by a *big mutation*: each individual in these generations mutates $\sqrt{L}$ times.
5. K+KN: this is similar to 1+N with the exception that each generation consists of $K = \sqrt{L/4}$ individuals. In this work K equals to 5.

Thus, we applied 25 different scheduling algorithms: each strategy for each model. Each scheduler is called by its strategy.

3.2 Experiment setup and test data

We build up artificial problem instances as it is a common practice for such type of problems. Each problem is defined by a set of points located inside a square 100×100.

We conducted two series of experiments, in which points where generated by two different probability distributions. In the first series the points are generated with the Gaussian distribution with centre (50, 50) and variance equal to 12.5. In the second series the points are generated with the uniform distribution.

As it was stated in [11], $12 \cdot L^2$ application of small moves is enough and the route is unlikely to change anymore, because the system reaches a local optimum. In our case, $12 \cdot L^2$ equals to 30,000. However, we held experiments for 200,000 fitness function evaluation.

Each of the described small moves was tested on each of the two experiment series 100 times: 10 times on 10 different problems.

For each run, we measured the *relative improvement* of the found solution in comparison to the initial naïve solution: $(i_1, i_{L+1}, i_2, i_{L_2}, \ldots, i_L, i_{2L})$.

In order to compare different algorithms, we applied Wilcoxon signed rank test: since all the algorithms were run 10 times on 10 datasets. Thus, for each dataset we evaluated mean relative improvement, and then treated it as paired data.

3.3 Results

Results for problems generated with uniform distribution, are presented in Table 1. We chose the best result for each small move, after which we have compared this result with all other results using Wilcoxon signed rank test (see Table 2). Thus, we colored the best results for each small move with light grey, and the best results for all small move — with dark grey.

We can see that the two last EAs outperformed all the other in most of the cases. However, the best results were achieved not solely by them, but also by (1+N). We see that the greedy strategy (1+1) is the worst one. Also, Double-Bridge small move shows the same results for all EA (which is relative poor).

Results for problems generated with Gaussian distribution, are presented in Table 3 and results for statistical comparison are presented in Table 4. Colorization notation remains the same.

We see that points generated with Gaussian distribution shares the same patterns as the previous case. The only difference is that (1+1) also reaches the best result.

We have plotted behavior for different algorithms. All the plots can be found in a hosting[1]. (1, N) is worst by discovering Figure 1: because (1, N) excludes the parent from

[1]http://genome.ifmo.ru/files/papers_files/GECCO2016/

	Small moves				
EA type	Lin-2-Opt	Couple-Exchange	Double-Bridge	Point-Exchange	Mixed
1+1	0.703 (0.031)	0.493 (0.031)	0.560 (0.030)	0.499 (0.030)	0.735 (0.029)
1+N	0.709 (0.028)	0.494 (0.039)	0.559 (0.030)	0.500 (0.032)	0.738 (0.025)
1, N	0.712 (0.028)	0.250 (0.043)	0.560 (0.030)	0.252 (0.044)	0.545 (0.047)
1+N+BM	0.714 (0.029)	0.501 (0.029)	0.559 (0.030)	0.507 (0.030)	0.736 (0.029)
K+KN	0.708 (0.027)	0.502 (0.030)	0.560 (0.030)	0.506 (0.029)	0.740 (0.024)

Table 1: Relative improvement: each cell contains the mean and standard deviation (in braces) improvement of the corresponding EA–Small moves pair for points generated with uniform distribution. The best results for each small move are colored with grey. The best results are colored with dark grey.

	Small moves				
EA type	Lin-2-Opt	Couple-Exchange	Double-Bridge	Point-Exchange	Mixed
1+1	0.005	0.003	0.323	0.011	0.018
1+N	0.102	0.008	0.401	0.003	0.224
1, N	0.038	0.003	0.323	0.003	0.002
1+N+BM	BEST	0.401	0.480	BEST	0.084
K+KN	0.011	BEST	BEST	0.255	BEST

Table 2: P-values for Wilcoxon pair-rank test: comparison with the best scheduler for points generated with uniform distribution. P-values greater than 0.05 are colored with strong grey.

	Small moves				
EA type	Lin-2-Opt	Couple-Exchange	Double-Bridge	Point-Exchange	Mixed
1+1	0.657 (0.027)	0.460 (0.027)	0.525 (0.024)	0.467 (0.026)	0.693 (0.022)
1+N	0.659 (0.030)	0.462 (0.027)	0.526 (0.025)	0.466 (0.024)	0.697 (0.025)
1, N	0.675 (0.023)	0.244 (0.039)	0.524 (0.025)	0.248 (0.037)	0.517 (0.040)
1+N+BM	0.670 (0.030)	0.470 (0.025)	0.527 (0.023)	0.474 (0.023)	0.695 (0.021)
K+KN	0.670 (0.028)	0.475 (0.024)	0.525 (0.025)	0.477 (0.022)	0.695 (0.025)

Table 3: Relative improvement: each cell containts the mean and standard deviation (in braces) improvement of the corresponding EA–Small moves pair for problems generated with Gaussian-distribution. The best results for each small move are colored with grey. The best results are colored with dark grey.

	Small moves				
EA type	Lin-2-Opt	Couple-Exchange	Double-Bridge	Point-Exchange	Mixed
1+1	0.003	0.003	0.166	0.003	0.070
1+N	0.011	0.003	0.440	0.003	BEST
1, N	BEST	0.003	0.038	0.003	0.003
1+N+BM	0.166	0.006	BEST	0.142	0.102
K+KN	0.069	BEST	0.288	BEST	0.255

Table 4: P-values for Wilcoxon pair-rank test: comparison with the best scheduler for points generated with Gaussian-distribution. P-values greater than 0.05 are colored with strong grey.

the new generation, we see this behavior prevents solution improvement.

Also, visual analysis shows that the algorithms reach its highest values relatively early, as an example, see Figure 2. Thus, for each algorithm we discovered, when it reaches its best results. This comparison for the mixed small move is presented in Table 5. We see, that K+KN is relatively fast for both types of the data.

	EA type				
Problems	1+1	1+N	1, N	1+N+BM	K+KN
Uniform	161	158	196	146	122
Gaussian	158	113	86	183	131

Table 5: Number of thousands of steps, taken by each EA using mixed small move to find the optimum.

4. CONCLUSION

In this paper, we have proposed an evolutionary algorithm based solution for the Single Vehicle Pickup and Delivery problem. By using the small moves proposed in [9] as mutation operators for evolutionary algorithms, we were able to sufficiently increase the efficiency of SVPDP solution. According to the obtained results, we may state that (K+KN) EA is the best scheduler in the terms of speed and resulting performance. The commonly used greedy strategy was outperformed by the most of the applied algorithms.

In the future, we plan to develop *strategies* for small moves application. A strategy is a vector where each element defines the probability of selecting a particular small move. Strategy construction can be carried out using reinforce-

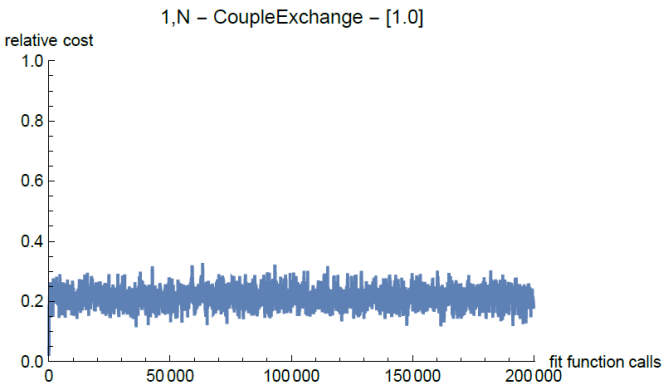

Figure 1: Plot for 1, N for Couple Exchange small move, dependency on of relative improvement on number of the fitness function evaluation

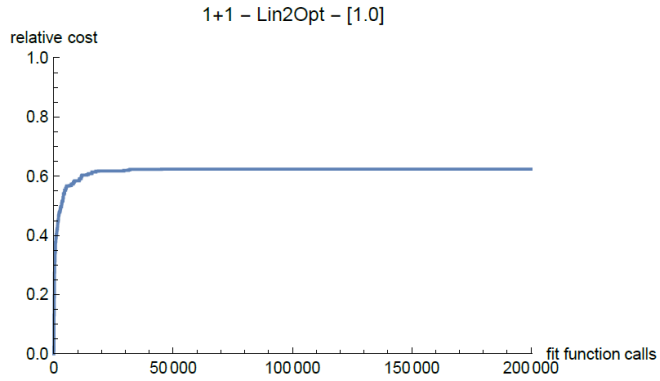

Figure 2: Plot for 1+1 for Lin2Opt small move, dependency on of relative improvement on number of the fitness function evaluation

ment learning, initial strategy selection can be done with meta-learning.

Furthermore, it may be beneficial to learn a Markov chain or a nondeterministic finite automaton that would define transitions between vectors. The insight behind this is that a smart application of small move series may be beneficial.

Finally, in the long perpective, we plan to apply meta-learning on the level of solutions rather than on the level of the problem. This way we would calculate some meta-features for each solution and form the strategy vector on this basis.

Acknowledgements

Authors would like to thank Ivan Smetannikov for technical support and Lidia Perovskaya for useful comments. This work was financially supported by the Government of Russian Federation, Grant 074-U01.

5. REFERENCES

[1] F. Carrabs, J. Cordeau, and G. Laporte. Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading. *INFORMS J. on Computing*, 19(4):618–632, 2007.

[2] B. Cheang, X. Gao, A. Lim, H. Qin, and W. Zhu. Multiple pickup and delivery traveling salesman problem with last-in-first-out loading and distance constraints. *European journal of operational research*, 223:60–75, 2012.

[3] D. Chivilikhin and V. Ulyantsev. Muacosm: A new mutation-based ant colony optimization algorithm for learning finite-state machines. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 511–518. ACM, 2013.

[4] J. Cordeau, M. Iori, G. Laporte, and J. Salazar GonzÃąlez. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with lifo loading. *Networks*, 55:46–59, 2010.

[5] J. Cordeau, G. Laporte, and S. Ropke. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Recent Models and Algorithms for One-to-One Pickup and Delivery Problems, pages 327–357. Springer US, 2008.

[6] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[7] M. Iori and S. Martello. Routing problems with loading constraints. *Top*, 18:4–27, 2010.

[8] D. Johnson and L. McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997.

[9] Q. Lu and M. M. Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672–687, 2006.

[10] H. Pollaris, K. Braekers, A. Caris, G. Janssens, and S. Limbourg. Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, 37:297–330, 2015.

[11] J. Schneider and S. Kirkpatrick. *Stochastic optimization*. Springer Science & Business Media, 2007.