Improving the Initialization and Repair Heuristics to Effectively Solve the Pickup and Delivery Problems with Time Windows

Vincent Tam and **M.C. Kwan** [1]

Abstract. Pickup and delivery problems with time windows (PDP-TW) are challenging yet realistic scheduling problems in which each delivery vehicle is assigned to handle different pairs of pickup-and-delivery requests. Our previous work successfully adapted the push forward insertion heuristic (PFIH) for initialization, and proposed a new swap operator to effectively solve PDP-TWs. In this paper, we improved our adapted PFIH to efficiently obtain a better initial solution. Moreover, we proposed an adaptive swap (*AD-Swap*) operator that can flexibly revise its neighborhood size for iterative improvements. Our improved search prototype achieved remarkable results against those of a tabu-embedded metaheuristic search or our previous work on a set of modified Solomon's test cases. More importantly, our improved heuristics can easily be integrated into many search schemes to solve other scheduling problems.

1 INTRODUCTION

In both Artificial Intelligence [1, 3] and Operations Research [2, 5], delivery problem with time windows (DP-TW) notably represents a class of challenging delivery problems with a wealth of published results [1, 5, 6]. The main aim is to effectively schedule a fleet of delivery vehicles in order to satisfy a number of customers' requests with user-specified service time windows, thus restricting each delivery to occur within a limited period. Extending from DP-TWs, the pickup and delivery problems with time windows (PDP-TW) [6], with additional coupling constraints to request each pair of pickup and delivery requests to be serviced by the same delivery vehicle, represent a more general and challenging class of delivery problems with wider applicability to modern logistics applications for the land, sea or air transport. Examples of PDP-TWs include the dial-a-ride application and bus scheduling. Heuristics have been widely used for vehicle routing due to their effectiveness. The push forward insertion heuristic (PFIH) [5] was a route construction heuristic proposed by Solomon to handle DP-TW. Basically, PFIH compares the cost of inserting a new customer into the current route with the lowest possible cost against that of creating a new route. A modified PFIH was proposed in [6] to handle PDP-TW.

For vehicle routing, many local search methods like the tabu search [2] or genetic algorithms [1] often use the PFIH or its variants to construct an initial solution before applying any heuristic operators to optimize the objective value of the current solution. Tabu search (TS) [2] is an example search strategy, with the use of short-term memory to avoid cycling, to solve many practical combinatorial problems including timetabling [3], integrated circuit design and vehicle routing [2]. Besides, genetic algorithm (GA) [1] is a type of adaptive heuristic search based on natural evolution. A population of chromosomes is generated and continuously modified by some genetic operators to produce offsprings for another new generation until a predefined stopping criterion is reached, or a locally optimal solution is found. An example of GA is the GIDEON [1] algorithm.

In a previous work [7], a min-conflicts based micro-genetic algorithm [1, 7] was adapted to solve PDP-TW. Moreover, two interesting initialization heuristics namely the Align-Fold and Boomerang, as alternatives to the adapted PFIH, together with a new Swap operator for repairing the current solution were proposed to effectively solve the PDP-TW. The proposed initialization and repair heuristic operators were later integrated into 6 different search algorithms, and obtained impressive results on a set of modified Solomon's test cases. In this paper, we firstly review the Align-Fold and adapted PFIH initialization methods, and then improve the adapted PHIF initialization heuristic to aggressively look for a better initial solution in an efficient manner. In addition, we proposed an adaptive swap (*AD-Swap*) operator that can flexibly adjust its neighborhood size most suitable for iterative improvements. Obviously, our *AD-Swap* operator can be regarded as a reduced version of the large neighborhood search (LNS) method described in [4]. After all, our improved search prototype achieved remarkable results against those of a tabu-embedded metaheuristic search [2, 6] or our previous work on the set of modified Solomon's test cases. More importantly, our improved initialization and repair heuristics were so generic and thus easily integrated into many search schemes to possibly solve other optimization problems.

The paper is organized as follows. Section 2 reviews the basic concepts and definitions about the PDP-TW. Section 3 considers the original and improved versions of the initialization and repair heuristics. Section 4 gives an empirical evaluation on the performance of our improved heuristic search method against those of the original search method and Li & Lim's metaheuristic approach [6]. Lastly, we conclude our work in Section 5.

2 PICKUP AND DELIVERY PROBLEMS WITH TIME WINDOWS (PDP-TW)

Similar to DP-TW, pickup and delivery problems with time windows (PDP-TW) are constrained optimization problems [3]. The formal definition of PDP-TW [6, 7] is stated as follow. Given a node set $N = \{n_0, n_1, n_2, n_3, \ldots, n_m\}$ where n_0 always denotes the depot, n_1 to n_m denote delivery or pickup locations for customers' requests,

[1] Dept. of E.E.E., The University of Hong Kong, Pokfulam, Hong Kong. email: vtam@eee.hku.hk

and the last index m is always an even number, each individual customer request is represented by a pair of delivery and pickup locations. Each delivery or pickup location n_i where $i \neq 0$ is associated with a customer demand q_i such that $q_i > 0$ for a pickup location whereas $q_i < 0$ for a delivery location, a service time s_i, that is the duration required to effectively service the customer demand at that location, and an associated service time window $[e_i, l_i]$ where e_i and l_i denote the earliest and latest time to start the service. The delivery and pickup demand q_i and q_j belonging to the same customer will have the same magnitude so that $q_i + qj = 0$ for ease of analysis. Besides, for any possible edge $\langle n_i, n_j \rangle$, both the non-negative distance d_{ij} and required travel time t_{ij} are specified. However, it should be noted that due to the time-window constraints, not every possible edge is a feasible edge to construct a feasible route for any vehicle when solving the DP-TW or PDP-TW. In other words, only those edges $\langle n_i, n_j \rangle$ that satisfy their corresponding time-window constraints as $t_{oi} + s_i + t_{ij} \leq l_j$, restricting the vehicle concerned to arrive at or before the latest service time l_j after traveling from the depot to n_i to n_j with its completion of service at n_i, should be considered.

In addition to the time-window constraints, several problem constraints must not be violated. First, each vehicle has a limited capacity C that cannot be exceeded. Each vehicle must carry an amount less than or equal to C. Second, all vehicles depart from and return to the same depot n0, and share the same constraints time window $[E, L]$, where E denotes the time a vehicle must have left the depot, and L denotes the time a vehicle must have returned to the depot. Third, a customer can only be serviced within the associated service time window $[e_i, l_i]$. That is, if a vehicle reached the customer earlier than e_i, the vehicle has to wait until e_i. Lastly, the coupling constraints request that every pair of pickup and delivery locations must be serviced by the same vehicle while the precedence constraints specify that the pickup location must be serviced first. Clearly, the objective functions vary depending on different applications. For instance, in the dial-a-ride application, a common objective is to minimize the inconvenience (often measured in term of the total waiting time) as caused by the service to occur earlier or later than the expected time. Following [6], we consider in this paper an objective cost function with 4 parameters in descending order of importance as follows: the number of vehicles used, the total traveling cost, the total schedule duration, and the drivers' total waiting time.

3 IMPROVED HEURISTICS TO SOLVE PDP-TWs

Generally speaking, heuristics play a very significant role in affecting the overall performance of a local search method. When handling the DP-TW or PDP-TW, heuristic operators can be classified as intra-route or inter-route operators [7] depending on whether the operator works within or between the route(s). Clearly, due to the 'coupling' nature of delivery and pickup nodes in PDP-TW, most heuristics operators designed for DP-TWs need to be slightly modified to effectively tackle the PDP-TW. In the following subsections, we are going to firstly consider the adapted push forward insertion heuristic (PFIH) as originally applied in [6], followed by our proposed improvement to aggressively look for a better initial solution for solving PDP-TWs efficiently. Later, we would compare the original and improved Swap operators using a flexible revision scheme to adjust neighborhood size for iterative improvements.

3.1 Initialization heuristics

When handling PDP-TWs, with a sorted list of customer-pairs based on their combined distance from the depot, a local search algorithm can proceed to construct an initial solution possibly using one of our proposed initialization heuristics detailed as follows:

- **The Align-Fold Initialization Method**
  The initialization heuristic involves 2 phases: Align and Fold. In the first Align stage, customer pairs are inserted into a vehicle one by one from the sorted list of customer pairs in descending order of their combined distance from the depot. When a customer pair cannot be inserted into the current route due to constraints violation, a new route will be created. The insertion process continues until all customer pairs are routed. After the Align Stage, there may be gaps of different sizes in the beginning few routes, that can possibly be filled up by customer-pairs (of smaller distance from the depot) shifted from the last few routes. Such shifting and filling operation is called folding. Folding requires a boundary line to be set among the vehicles. Following the strategy to preset the number of "virtual" vehicles for constructing the initial solution in [9], we refer to research results produced by Li and Lim [7] to determine the boundary line for each case. For each customer-pair to be folded, we compute the folding cost for every possible insertion position in any upper vehicle i as $i \times (scheduled\ duration\ per\ route\ + distance\ traveled\ by\ the\ vehicle\ per\ route)$. The position with the least folding cost would be chosen. Obviously, we will iteratively try for more folding at a higher boundary line whenever possible.

- **The originally adapted PFIH**
  The push forward insertion heuristic (PFIH), originally designed for DP-TWs, was successfully adapted [6] to tackle PDP-TWs. The adapted PFIH basically considers each customer pair as a single unit of requests to be handled by every delivery vehicle. The adapted algorithm is clearly specified in pseudo-code as follows.

  (1) Start a new route to insert a "customer pair" taken from the beginning of a customer-pairs list sorted in decreasing order of their combined objective values (as described in Section 2);

  (2) Remove a customer pair from the sorted customer-pairs list **while** the list is not empty. After evaluating *all* feasible positions of *current* route, insert the newly removed customer pair into a position of the *current* route which cause the least increment in the combined objective value;

  (3) If the customer pair cannot be assigned to any position of the *current* route, starts a new route and assign the customer pair to this newly created route;

  (4) Repeat step (2) and (3) **until** all customer pairs are assigned.

- **The improved best-fit PFIH**
  One obvious shortcoming of the originally adapted PFIH method is that it does not even guarantee to return a locally optimal solution with respect to all the available routes during the search. Accordingly, we decided to improve the adapted PFIH to aggressively look for the best-fit positions among all the existing routes for each customer pair for insertion with the following modifications to step (2) and (3) of the previous adapted PFIH algorithm:

  (2*) Remove a customer pair from the sorted customer-pairs list **while** the list is not empty. After evaluating *all* feasible positions of *all* route, insert the newly removed customer pair into the best-fit position of any route which cause the least increment in the combined objective value;

(3*) **If** the customer pair cannot be inserted to any existing route, starts a new route and assign the customer pair to this newly created route;

Since our proposed improvement here actively examines all existing routes for any better improvement, the quality of most initial solutions generated by our improved best-fit PFIH (BPFIH) will always outperform the solution quality generated by the original adapted PFIH. However, the major concern is whether our proposed BPFIH can be efficient for execution. In fact, in a preliminary evaluation of the BPFIH, we already found that our implemented BPFIH could always return an initial solution in less than 2 (wall clock) seconds, almost the same time required by the original adapted PFIH, for any of the 56 modified Solomon's test cases of PDP-TWs running on an Intel P4 2.8 GHz desktop computer. However, over these 56 tested PDP-TWs, the overwhelming initial solution quality was guaranteed by our improved BPFIH with the total number of vehicles used as 549 and the total distance traveled as $101,999.95$ standard units, such as km, whereas the figures obtained by the original adapted PFIH were 924 and $148,405.96$ units respectively. This drastic difference clearly indicates that our improved BPFIH can be implemented and executed efficiently to obtain a much better initial solution as compared to that of the originally adapted PFIH.

3.2 Repair heuristics

In the following, we consider two interesting repair heuristics, namely the original and improved $Swap$ operators, to iteratively improve the objective value of the current routing plan until a local minimum is reached.

- **The original $Swap$ operator**
  The original $Swap$ operator works by a substantial modification of the current solution. From each vehicle in the fleet, the $Swap$ operator will randomly pick up a few pairs of customers, remove them from the vehicle and add them into a relocation pool. The number of customer pairs to be removed from each vehicle depends on fleet size. The major consideration is to keep the relocation pool having roughly the same number of customers in each swap. As an example, in our prototype implementation, we arbitrarily set the Swap operator to remove around $1/5$ of total number of customers into the relocation pool. When there is any empty vehicle after removal of customers, the vehicle would be removed from the fleet. Then, the operator will randomly choose a pair of customers from relocation pool and insert them into any vehicle based on the objective cost function described in Section 2. In case there is no possible position in any route to insert the customer pair(s) in the relocation pool, a new vehicle would be created. After all, the $Swap$ operator is closely related to the large neighborhood search (LNS) method [4], and can be viewed as a reduced version of LNS, thus taking up less computational resource, to repair the current solution for solving PDP-TWs efficiently.

- **The improved $AD\text{-}Swap$ with adaptive reallocation pool size**
  The previous $Swap$ operator suffers from two major drawbacks. First, it is generally very difficult to determine a "suitable" relocation pool size in each iteration for the same/different problems. Second, the $Swap$ operator is applied only once per search iteration, and may sometimes create a poorer solution with a larger number of vehicles used. In fact, the $Swap$ operator can be more appropriately used for iterative improvements in which the $Swap$ operator can be applied several times per search iteration to opportunistically look for a better solution with a flexibly adjustable relocation size. The result is an adaptive $AD\text{-}Swap$ operator as proposed in this paper.

Undoubtedly, both $Swap$ and $AD\text{-}Swap$ *randomly* extract customer-pairs from the existing solutions for re-insertion to look for opportunistic improvements. However, other than the major difference in the reallocation pool size, we should carefully consider the following difference in using the original $Swap$ against the improved $AD\text{-}Swap$ to solve PDP-TWs. First, the extracted customer-pairs list will *always* be sorted in a decreasing order of the combined distance traveled in the original $Swap$ whereas the same list will be *alternatively* sorted and unsorted in the improved $AD\text{-}Swap$ so as to allow more flexibility and thus possible improvements during the search. Moreover, whenever the number of vehicles required exceeds that of the currently best solution, a new vehicle will still be created, and the search continues in the original $Swap$ whereas the search will be simply halted and restarted in the improved $AD\text{-}Swap$. In other words, the number of vehicles required during the search is "bounded" by that of the best solution found in the improved $AD\text{-}Swap$ operator.

In addition to the above $AD\text{-}Swap$ scheme, we try a search strategy opposite to iterative deepening (ID) [7] so as to effectively control the resources used to explore the possible improvements during the search. Instead of iteratively increasing the resource limit to explore various sub-trees in ID, we gradually reduce the relocation pool size over iterations in order to minimize the computational overheads involved in the $AD\text{-}Swap$ scheme. The final result is the Iterative Diminishing Swap ($ID\text{-}Swap$) operator that is simply a variant of the $AD\text{-}Swap$ operator using a strategy to gradually diminishing the relocation pool size.

4 AN EMPIRICAL EVALUATION

To demonstrate the effectiveness of our proposed improvement, our improved initialization and repair heuristics were integrated into the previous local search framework [7] as an improved optimizer named ($BPFIH + ID\text{-}Swap$) to compare against the original optimizer ($PFIH{+}Swap$) [7], another local optimizer ($Align{+}Swap$) using the Align-Fold initialization method and the $Swap$ operator, and Li & Lim's metaheuristic approach [6] integrating tabu search and simulated annealing on a set of 56 modified Solomon's test cases [6]. Basically, each problem instance has around 100 customers. There are totally 6 distinct classes, namely LC1, LC2, LR1, LR2, LRC1, and LRC2. 'LC' refers to cases with clustered distribution of customers, 'LR' refers to cases with uniform distribution of customers, and 'RC' refers to mixed customers types. '1' refers to small vehicle capacity whereas '2' refers to large vehicle capacity.

All of our search prototypes were implemented using the Visual C++ Version 7.0. And all tests were run on a desktop computer with Intel Pentium IV processor of 2.8 GHz, 512 MB RAM, and a hard disk of 20 GB space. The operating system used was the Microsoft Windows XP. It should be noted that the original ($PFIH{+}Swap$) optimizer with a constant relocation pool size of 10 customer pairs would halt after no improvement over 30 consecutive iterations whereas our improved ($BPFIH + ID\text{-}Swap$) optimizer would keep on adjusting its relocation pool size from 35 pairs with a decrease of 5 pairs after every 240 iterations. And the improved optimizer would stop after a total of $1,440$ iterations.

Table 1 summarizes the overall results, in terms of the total number of vehicles used ($\sum TV$), the total distance traveled ($\sum TD$) and the sum of their products as $\sum(TV \times TD)$, of the different search proposals over all the 56 modified cases. The smallest figure in each column was boldfaced for ease of comparison. Clearly,

Optimizers	Overall Results		
	$\sum TV$	$\sum TD$	$\sum(TV \times TD)$
$Align + Swap$	415	$58,743$	$479,974$
$PFIH + Swap$	417	$58,410$	$481,426$
$BPFIH + ID\text{-}Swap$	410	**57,766**	$467,197$
Li & Lim's approach	**405**	$58,185$	**462,873**

Table 1. Overall results of different optimizers on all 56 modified benchmarks

the $(BPFIH + ID\text{-}Swap)$ optimizer overwhelmingly excelled the original $(PFIH + Swap)$ and $(Align + Swap)$ optimizer on both TV and TD of overall results revealed in Table 1, demonstrating the effectiveness of our proposed improvements over the original heuristics used. Besides, Li & Lim's metaheuristic approach achieved the best $\sum TV$ and $\sum(TV \times TD)$ results while our proposed $(BPFIH + ID\text{-}Swap)$ optimizer obtained the best result in $\sum TD$. The $(Align + Swap)$ optimizer achieved the second best results among those of our heuristic search proposals. It should be noted that the Li & Lim's metaheuristic approach is very complicated to implement with lots of parameter tuning. Yet it only excelled our overall result in $TV \times TD$ slightly by 3% less. On the other hand, our direct heuristic approach of the $(BPFIH + ID\text{-}Swap)$ or $(Align + Swap)$ optimizer is easy to implement.

Our optimizers against Li & Lim's approach	Performance statistics		
	Win	Loss	Tie
$Align + Swap$	5	30	21
$PFIH + Swap$	3	19	34
$BPFIH + ID\text{-}Swap$	**9**	9	**38**

Table 2. Comparing our proposed optimizers against Li & Lim's results

Table 2 gives a direct comparison of our proposed $(Align + Swap)$, the original $(PFIH + Swap)$ and refined $(BPFIH + ID\text{-}Swap)$ optimizer against Li & Lim's metaheuristic approach based on their individual results obtained on each test case. In terms of number of wins, ties and loses, our refined $(BPFIH + ID\text{-}Swap)$ optimizer was surely the best among our search proposals with 9 results better than those of Li & Lim's published results over the 56 test cases. The overall improvement could be explained by a better initial solution generated by more ambitious BPFIH method and the refined $ID\text{-}Swap$ operator that aggressively looked for the greatest improvements initially with a larger relocation pool size, and then gradually decreasing the relocation size for opportunistic improvements in the later stage.

Table 3 shows the detailed improvement of the $(BPFIH + ID\text{-}Swap)$ optimizer over Li & Lim's metaheuristic approach on the 9 specific test cases of PDP-TW. Among the 9 winning cases as shown in Table 3, there were more significant decreases in TD as obtained by our improved $(BPFIH + ID\text{-}Swap)$ optimizer on the

Test Case	$BPFIH + ID\text{-}Swap$		Li & Lim's approach	
	TV	TD	TV	TD
LC204	3	**590.60** ($\downarrow$ 0.10%)	3	591.17
LR201	4	**1253.23** ($\downarrow$ 0.84%)	4	1263.84
LR209	3	**930.59** ($\downarrow$ 0.69%)	3	937.05
LRC106	11	**1424.74** ($\downarrow$ 0.06%)	11	1425.53
LRC107	11	**1230.14** ($\downarrow$ 0.0008%)	11	1230.15
LRC108	10	**1147.43** ($\downarrow$ 0.05%)	10	1147.97
LRC201	4	**1406.94** ($\downarrow$ 4.22%)	4	1468.96
LRC204	3	**818.66** ($\downarrow$ 1.10%)	3	827.78
LRC206	3	**1159.03** ($\downarrow$ 0.33%)	3	1162.91
LRC207	3	**1062.05** ($\downarrow$ 25.45%)	3	1424.60

Table 3. Detailed improvement of our $(BPFIH + ID\text{-}Swap)$ optimizer over Li & Lim's results on the 9 winning cases

mixed LRC problems with larger vehicle capacity that flexibly allows greater opportunity for improving the solutions after swapping certain customer pairs. On the other hand, as revealed in Table 2, there were other 9 cases where our improved optimizer lost to Li & Lim's approach in the TV or TD results. However, it should be noted that our improved optimizer was simple and thus easy to implement/tune as compared to Li & Lim's metaheuristic approach integrating both simulated annealing and tabu search that were undoubtedly more complicated to implement/tune. Furthermore, all the published results obtained by Li & Lim's metaheuristic approach were already very much minimized and likely near-optimal as achieved through their fine-tuned annealing schedule of the embedded simulated annealing optimizer and its complex interaction with the tabu search. Therefore, it is in fact very difficult to further improve any of their published results on the set of 56 modified test cases.

5 CONCLUDING REMARKS

In this paper, a formal definition [7] of the pickup and delivery problems with time windows (PDP-TW) was given as a more general and challenging class of delivery problems. We proposed several interesting initialization and repair heuristics, including the best-fit push forward insertion heuristic (BPFIH) and the adaptive repair-based $AD\text{-}Swap$ operator to improve on the originally adapted push forward insertion heuristic (PFIH) [10] and the previous $Swap$ operator. In general, our obtained results compared favorably against those of Li & Lim's tabu-embedded metaheuristic search proposal and a previous work [7] on $Swap$ on a set of modified Solomon's test cases. Specifically, our $(BPFIH + ID\text{-}Swap)$ optimizer outperformed Li & Lim's approach on 9 test cases. This clearly demonstrates the further optimizing capability of our improved $Swap$ operator over the other heuristic approaches we have considered in this paper.

There can be many interesting directions for future investigations. Examples include the investigation about the iterative-increasing Swap $(INCR\text{-}Swap)$ operator opposite to the iterative diminishing resource allocation scheme. Besides, applying these improved heuristics to solve other scheduling problems like the original delivery problems with time windows should be interesting.

ACKNOWLEDGEMENTS

We would like to thank the referees for their valuable comments. In addition, we are grateful to Miss Lois Tseng who implemented the initial prototype of the $(Align + Swap)$ optimizer used for experimentation.

REFERENCES

[1] Braysy, O.: Genetic Algorithms for the Vehicle Routing Problem with Time Windows. Special issue on Bioinformatics and Genetic Algorithms, Arpakannus 1/2001.

[2] Glover, F.: Tabu Search - Part I. ORSA Journal on Computing, **Volume 1, Number 3**, Summer, 1989.

[3] Holland, J.: Adaptation in Naturaland Artificial Systems. University of Michigan Press, Ann Arbor, 1975.

[4] Jee, J.: Solving Vehicle Routing Problems with Time Windows using Micro-Genetic Algorithms. Undergraduate Research Opportunity Project (UROP) report, School of Computing, The National University of Singapore, 1999/2000.

[5] Kancko, K., Yoshikawa, M., Nakakuki, Y.: Improving a Heuristic Repair Method for Large-Scale School Timetabling Problems. Principles and Practice of Constraint Programming - CP99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999.

[6] Lau, H., Liang, Z.: Pickup and Delivery Problems with Time Windows: Algorithms and Test Case Generation. in *Proceedings* of the 13th IEEE International Conference on Tools with Artificial Intelligence, Nov 7-9, 2001.

[7] Li, H., Lim, A.: A Metaheuristic for the Pickup and Delivery Problem with Time Windows. in *Proceedings* of the 13th IEEE International Conference on Tools with Artificial Intelligence, Nov 7-9, 2001.

[8] Minton, S., Johnston, M., Philips, A., Laird, P.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence, 1992.

[9] Prosser, P., Shaw, P.: Study of greedy search with multiple improvement heuristics for vehicle routing problems. Technical Report, **RR/96/201**, Department of Computer Science, University of Strathclyde, Glasgow, January 1997.

[10] Solomon, M.M.: Algorithms for the vehicle routing and scheduling problem with time windows. Operations Research, **35:254-265**, 1987.

[11] Tam, V., Tseng, L.: Effective Heuristics to Solve Pickup and Delivery Problems with Time Windows. in *Proceedings* of the 15th IEEE International Conference on Tools with Artificial Intelligence, Nov 3-5, 2003.

[12] Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, 1993.