Robust Solutions for Vehicle Routing Problems via Evolutionary Multiobjective Optimization

R. Scheffermann, M. Bender, A. Cardeneo

Abstract— **In many practical applications it is observable that optimal solutions are vulnerable to changes in environmental- or decision-variables and therefore become suboptimal or even infeasible in uncertain environments. Solutions immune or less vulnerable to such uncertainties are called robust. In this paper we present and compare two algorithms for creating robust solutions to the vehicle routing problem with time-windows (VRPTW) in which travel times are uncertain. In the first approach robustness is defined as a dedicated optimization objective and the NSGA2 algorithm is used to solve the VRPTW as a multi-objective optimization problem. A Pareto-front is generated that displays the trade-off between robustness and the total distance to be minimized. A second approach uses a modified predator-prey algorithm, that implicitly takes robustness into account by defining different travel-time-matrices for each predator. It can be shown that the predator-prey approach is much faster than the NSGA2 and still delivers viable results.**

I. Introduction

A. Motivation

The vehicle routing problem with time-windows (VRPTW) is a popular combinatorial optimization problem, mainly because it is immediately applicable to the real world problem of transporting goods from a single depot to multiple customers. When solved (close) to optimality, solutions frequently have the property of making maximal use of time-windows and vehicle capacities. In real world scenarios travel times between customers are highly uncertain due to uncertain weather-conditions, heavy traffic or other disturbances. As a result, when there is a delay on one or several roads, solutions become infeasible if a vehicle misses a customer's time-window. Practitioners are thus in need of algorithms that take possible disturbances into account and create robust solutions to transportation problems.

B. Literature Review

Sörensen and Sevaux [2] discuss the implication of non-deterministic data in vehicle routing and claim the need for robust and flexible solutions. Uncertainty in underlying data is examined in some prior work but mainly focuses on stochastic customer demand [3][15]. Stochastic travel times are covered only in relatively few publications. The first work dealing with stochastic driving and service-periods was published by Laporte, Loveaux and Mecure [4] in 1992. In their approach for each tour a maximal period is defined, which must not be excessed. A chance-constrained model is proposed and problems with up to 20 customers are solved with a branch-and-cut algorithm. Lambert, Laporte and Louveaux [5] use a savings algorithm to solve a VRP with up to 44 customers, also considering stochastic travel times. In a work by Kenyon and Morton [6] either the expected route duration or the probability of a route exceeding a preset duration is minimized. Russel and Urban [7] integrate Erlang-distributed tavel-times into a VRPTW and solve it defining penalty costs for delayed deliveries. A tabu-search algorithm is used to solve instances of up to 100 customers.

C. Document Structure

In our paper we first describe the problem and its underlying assumptions and definitions. We explain how robustness is understood in this work, how and why scenarios are used to evaluate solutions and what the main goal of this study is. The following two chapters introduce the two algorithms applied to the prior defined problem: the NSGA2 and the predator-prey algorithm. We then describe the experimental setup and present some results. The paper ends with a conclusion and an outlook to further work and suggests next steps.

II. Problem Description

A. Vehicle Routing Problem with Time-Windows

In the basic vehicle routing problem with time-windows (VRPTW) multiple customers have to be visited by vehicles starting and ending at a single depot (figure 1). Customers have known demands and vehicles have limited capacity. Travel times for each relation are given and customers have a single time-window in which the service has to start.

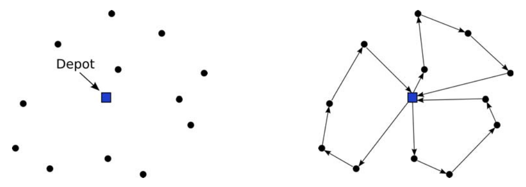

Fig. 1. VRPTW problem visualization and possible solution

To solve the problem several objectives may be taken into account – minimizing the number of vehicles needed, minimizing total distance travelled or minimizing total time needed to serve all customers (including waiting periods). A comprehensive description of the VRP and its variants can be found in [1].

Authors are with the FZI Forschungszentrum Informatik, Department of Logistics Systems Engineering, Haid-und-Neu-Strasse 10-14, 76131 Karlsruhe, Germany. For more information please contact Robert Schefferman at scheffer@fzi.de.

B. Scenarios

In our experiments the problem structure is derived from the well known Solomon-benchmark-instances [14]. The deterministic travel times t_{ij} (calculated as the Euclidean distance between customers' locations) are adjusted in our experiments to simulate fluctuations in driving times:

$$\hat{t}_{ij} = t_{ij} \cdot (1 + X) \tag{1}$$

Here t_{ij} is the best-case travel-time between customers i and j. X is a realization of a modified exponentially distributed random variable (as shown in figure 2), which can take values between 0 and 0.5 and represents the possible delay of up to 50%.

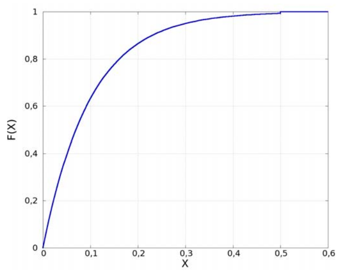

Fig. 2. exponential distribution with $\lambda = 10$

For our experiments scenarios are created using different travel-time matrices following equation 1. Therefore one problem-instance consists of different scenarios using different tavel-time-matrices but the same set of customers, same distances and same demands. As a result, a solution to one instance has a fixed number of vehicles and sequence and vehicle-assingment of visited customers. This solution is then evaluated via all scenarios resulting in different fitness-values due to different travel-times and possible violation of time-windows.

C. Robustness

Scholl [8] introduced several aspects of robustness in optimization and distinguishes between several types of robustness. Following his structure we are referring here to feasibility-robustness (a solution stays feasible in uncertain environments or becomes only slightly infeasible, which can for example be measured by penalty functions).

In our model delays (missing a time-window) have to be avoided, in other words a robust plan for a problem-instance will have few or no delays in a set of scenarios. Let $\delta_i(c, p)$ be the delay of a plan p at customer c in scenario i (with n being the number of scenarios created) then robustness $\rho(p)$ can be measured as the average delay,

$$\rho(p) = \frac{1}{n} \sum_i \sum_c \delta_i(c, p) \tag{2}$$

or, if our model should penalize longer delays above average, this can be integrated by taking the average squared delays:

$$\rho(p) = \frac{1}{n} \sum_i \sum_c \delta_i^2(c, p) \tag{3}$$

D. Research Objective

There are two main objectives in this work. One is to develop an easily adaptable model which is able to cope with the problem as realistically as possible. This is one reason for using scenarios – in many problems environmental variables are not statistically independent. In the VRPTW example if there is a traffic jam on one road, there will be traffic jam in the vicinity as well. Also, if there is bad weather causing a delay, there will be bad weather in the vicinity/whole region as well. Our model can easily be extended to cover a set of dedicated scenarios – also including the probability of realization for each scenario.

The second objective of this paper is to analyze the trade-off between robustness and solution-quality for the VRPTW: how great a detour must be accepted to improve the solution's robustness – this will be achieved by plotting and analyzing Pareto-fronts. Two algorithms are presented to calculate robust solutions: the Non-dominated Sorting GA (NSGA2) as a straightforward method considering robustness as a second optimization objective and a predator-prey-algorithm which takes robustness into account implicitly.

III. NSGA2 Approach

Evolutionary algorithms usually don't work on a single individual but on a population of several individuals. For this reason, evolutionary algorithms are suitable to solve multi-objective problems, as they are capable of creating multiple Pareto-optimal solutions in a single run [11]. For this reason they are a predestined method to solve the bi-objective VRPTW.

One of the most popular algorithms is the Nondominated Sorting Genetic Algorithm II (NSGA2) [9]. The NSGA2 is based on the following principles of Pareto-ranking, diversity control and elitism in selection.

A. Nondominated Sorting

At first solutions are evaluated by both fitness-functions (f_1, f_2) separately and plotted by their fitness-value in a 2-dimensional diagram. They are then arranged in Pareto-fronts by first determining all non-dominated solutions. These solutions are removed from the population and the procedure is repeated until all individuals are assigned to a Pareto-front.

Figure 3 shows the procedure: 14 solutions are assigned to three Pareto-fronts (assuming two objectives are to be minimized).

B. Crowding Distance

To achieve diversity in a population/front the crowding distance for each individual is calculated, which shows how close a solution is to other solutions of the same Pareto-rank. A solution with a high crowding distance is preferred over a solution with low crowding distance.

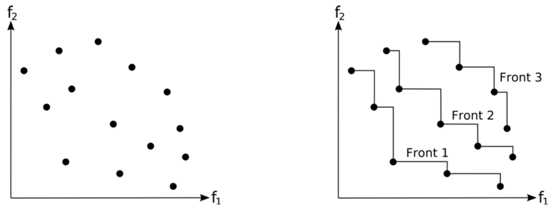

Fig. 3. Pareto-Ranking

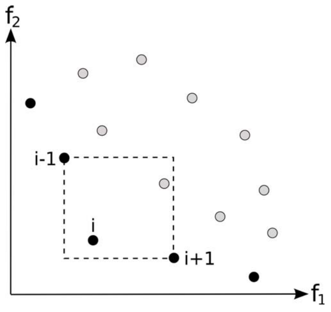

Fig. 4. calculating the crowding-distance for individual i

C. Selection of Individuals for further Generations

Figure 5 shows, how a population of children Q_t is created based on the parent-population P_t of size N using tournament-selection, mutation and/or recombination (Step I). In a next step all individuals are assigned to Pareto-fronts F_1 – F_6 (Step II). Fronts with low ranks (F_1 – F_2) are selected for the next generation P_{t+1} until the number of individuals in P_{t+1} would exceed N. If a front (F_3) can not be selected completely then for each individual in the population the crowding distance is calculated (Step III). The next generation is then filled up to N with those individuals having the highest crowding distance (Step IV).

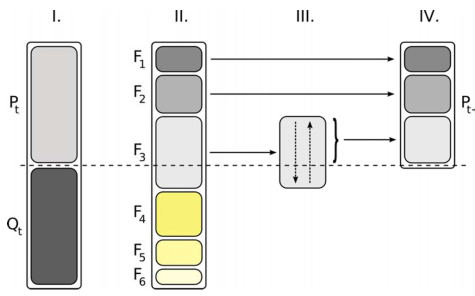

Fig. 5. NSGA2 selection of individuals for the next generation

After a stopping criterion (for instance a predefined number of generations) is met the algorithm stops with a set of Pareto-optimal individuals which displays the trade-off between the conflicting objectives.

D. Algorithmical Setup

Representation of individuals follow the Homberger and Gehring approach of 1999 [13]. An individual consists of three elements: a plan TP, a mutation instruction MV and a binary parameter RE.

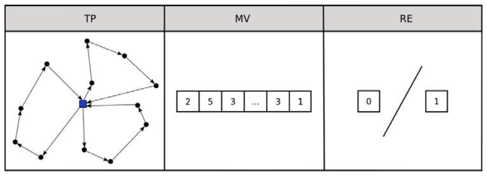

Fig. 6. Representation of individuals by Homberger and Gehring (1999)

For creating offsprings three individuals are selected randomly from the current parent-population. Two parents recombine their mutation instructions MV_1 and MV_2 to create their offspring's mutation instruction MV_3' following the uniform order-based crossover-operator by Davis [12].

The child consists of the third parent's plan TP_3 and parameter RE_3 and this mutation instruction MV_3', which is applied to the child as shown in figure 7.

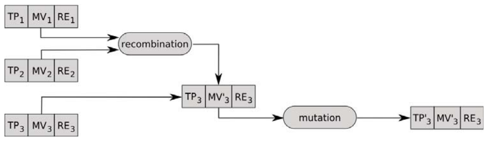

Fig. 7. Creating offsprings by Homberger and Gehring (1999)

The mutation instruction is a sequence of customers which contains each customer twice. This sequence is read from left to right. When a customer appears for the first time it is removed from the tour plan. When it appears for the second time it is reinserted following a greedy best insertion scheme. If the parameter RE is set to $true$, the algorithm tries to remove the shortest tour in a way that each customer inside of that tour is inserted into other tours if the solution is not rendered infeasible. For more details on the operator see [13].

In our model the two objectives are the total distance and the average (squared) delays as defined in section II-C. The population size N was set to 50 (for both parent population μ and children population λ), 200 scenarios were created using the exponential distribution presented in section II-B and the number of generations was set to 1500. In each generation each individual's distance was evaluated once (which is the same for all scenarios). To calculate the robustness a solution had to be evaluated using all 200 scenarios to calculate the average (squared) delays. So in a single run of the NSGA2 for 1500 generations 50 individuals had to be evaluated over 200 scenarios – resulting in 15.000.000 fitness-function evaluations.

While the application of the NSGA2 to the problem of finding robust solutions for the VRPTW and analyzing the trade-off between robustness and optimality is quite straightforward, it can be assumed that this high number of fitness-function evaluations makes the NSGA2 inefficient regarding its runtime, so the need for a faster algorithm arises which hopefully creates results of similar quality in shorter time.

IV. Predator-Prey Approach

The original Predator-Prey algorithm was published by Laumanns, Rudolph and Schwefel in 1998 [10]. It is a multiobjective evolutionary algorithm which does not follows the Pareto-dominance-paradigm and mimics the behaviour of predators hunting and consuming weak prey which creates a population of fit individuals.

The predator-prey-algorithm positions solutions (prey-individuals) on a grid. For each single objective a dedicated predator (predator = fitness function) is created and placed at a random position on the grid (so in a bi-objective problem there would be two predators). Now in one generation of the algorithm each predator evaluates the prey-individuals in his neighborhood and consumes the weakest prey (depending on the predator's fitness-function), which is then replaced by a mutation/recombination of the adjacent individuals. In the next round the predator moves on and evaluates the new neighborhood (figure 8) consuming another prey-individual. This way, individuals survive which are strong regarding all objectives, individuals with very low fitness in one of the objectives are consumed and replaced. In the end a population of individuals with good average fitness is obtained.

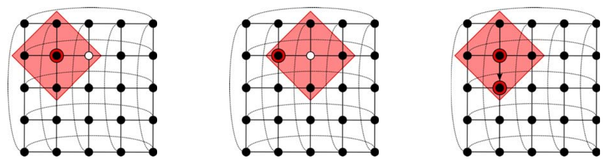
Fig. 8. Predator-Prey-Algorithm: a) find individual with lowest fitness b) consume and replace individual c) move on

A. Modifications to the Original Algorithm

In the presented approach the predator-prey-algorithm needs to be modified, as in this work predators do not correspond to objectives – all predators follow the same objective function using different environmental variables: they evaluate individuals using their dedicated travel-time matrix. In other words: the scenarios of the NSGA2 have become predators. The evaluation is twofold: the total distance travelled plus the (squared) delays at each customer, similar to the NSGA2 approach (akin to adding up the two NSGA2 objective values).

$$\text{fitness} = \text{total distance} + \sum \text{delay}^{2} \qquad (4)$$

B. Algorithmical Setup

For replacing individuals in principle the same mechanism for reproduction is used as in the NSGA2 algorithm. From the four neighboring individuals there are three new individuals generated (parents picked randomly) and the best of these new individuals replaces the consumed prey. The size of the torus containing the prey is set to 10x10 and the same 200 scenarios generated for the NSGA2 are used as 200 predators. As shown in figure 8 the neighborhood has a range of one and the step size of a predator for each move is one. The algorithm was run for 15 generations and in each generation each predator consumed three individuals (thus making three moves).

Predators generate only feasible individuals regarding their own fitness function but can not make sure that their created solution is feasible for other predators too. If a predator creates infeasible individuals a simple repair function (cheapest reinsertion) is applied to the solution. While the insertion part of the mutation obviously cannot create infeasible solutions, surprisingly a removal of a customer can. This is possible because the triangle inequality is not necessarily always feasible when the travel times on each edge is increased randomly. In addition to the needed repair function we defined that tours containing only one customer are always considered feasible.

V. Experiments and Results

Experiments were performed on the Solomon benchmark [14], which consist of C, R and RC instances as shown in figure 9.

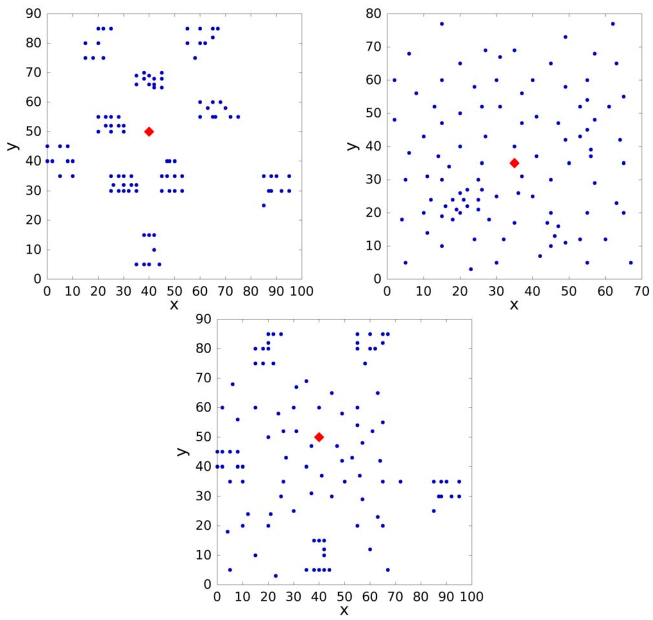

Fig. 9. Solomon's problem instances C101, R101 and RC101

In C-types customers are clustered, in R-types customers are randomly distributed and RC is a mixture of both. The benchmark has two different sets of instances, the type-1 instances have small time-windows and low vehicle capacities, the type-2 instances have wide time-windows and high vehicle capacities. In this paper, only results for the R1 and RC1 instances are discussed in detail. In type-2 instances there is not much room for optimization regarding robustness, even the solutions created with a single-objective algorithm have only very minor delays in a uncertain environment. In some C-type instances robust solutions are either nonexistent or have a significant increase in total driving distances. In type-2 instances there often was one dominating solution or there were only relatively few solutions in the first front. In this paper, due to space restrictions, we will visualize results for the R101 and RC101 instances only which are representative for all R1/RC1-types.

A. Comparison of NSGA2 and Predator Prey Results

As described in section III-D the NSGA2 was run for 1500 generations. The Pareto-fronts for the NSGA2 on the RC101 and R101 instances were calculated (figures 10, 11, 12 and 13) for both robustness key figures average delay and average squared delay. The calculation time was between 2075 seconds and 2172 seconds run on an Intel Pentium M 1,73 GHz CPU with 1GB RAM.

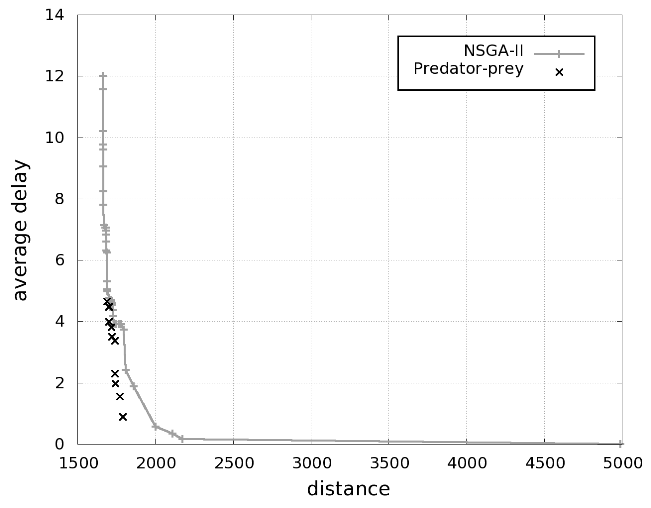

Fig. 10. NSGA-II vs. PP: R101 average delay

In all results for type-R1 and -RC1 instances it can clearly be seen that the Pareto-front is very steep and from the shortest solution found by the algorithm there is a major improvement in robustness for just a minor increase in total distance. Also – when just considering total distance – the best solution found is close to the best known solutions found in the literature (1645 for the R101 and 1696 for the RC101-instance). Although not directly comparable – as the heuristics used to create those results had minimization of vehicles as the prime objective – it shows that the overall quality of the solutions is in an acceptable range.

The predator-prey results can best be compared to the NSGA2 results by evaluating them the same way as the

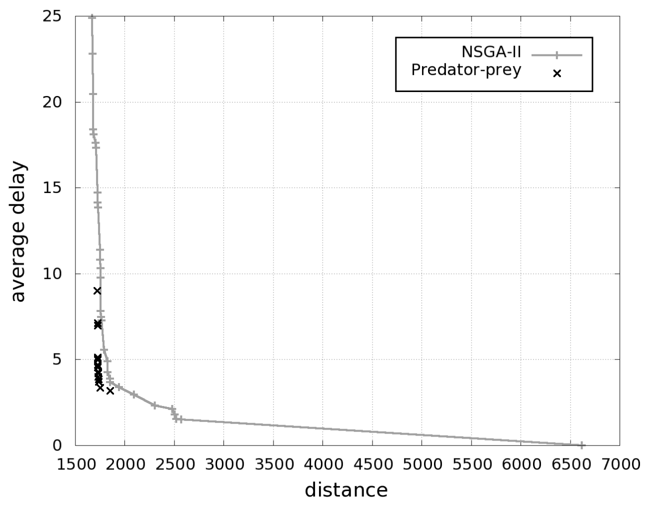

Fig. 11. NSGA-II vs. PP: RC101 average delay

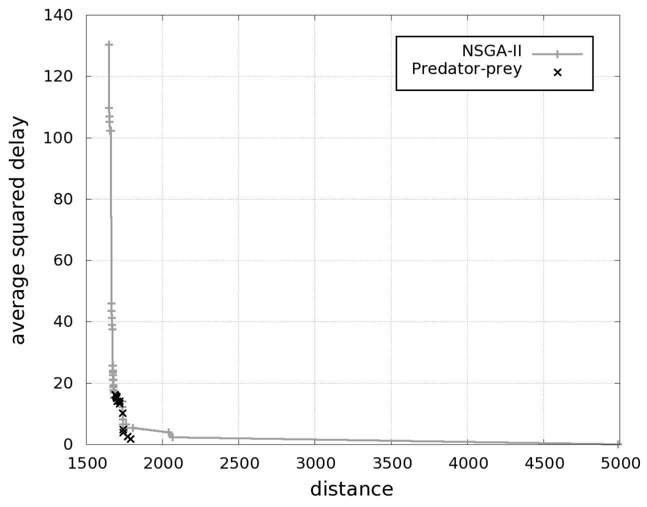

Fig. 12. NSGA-II vs. PP: R101 average squared delay

NSGA2-solutions, so after the predator-prey algorithm terminates, all solutions generated by the predator-prey algorithm are evaluated by all scenarios used in the NSGA2 and plotted into the same diagram as the NSGA2 results.

Compared to the NSGA2 the predator prey results cover only a small part of the Pareto-front. Still the solutions are mainly in the practically interesting part of the front, having a high robustness and only minor increases in total distance – often even dominating NSGA2 results in the same area.

B. Analysis of Predator Prey Results

To further analyze the predator-prey solutions, a single-objective version of the evolutionary algorithm was implemented to just minimize total distance. When the best (the median of sorted non-dominated solutions) predator-prey solution is compared to the best single-objective solution, it can be seen how much robustness can be achieved by accepting minor detours as shown in tables I and II.

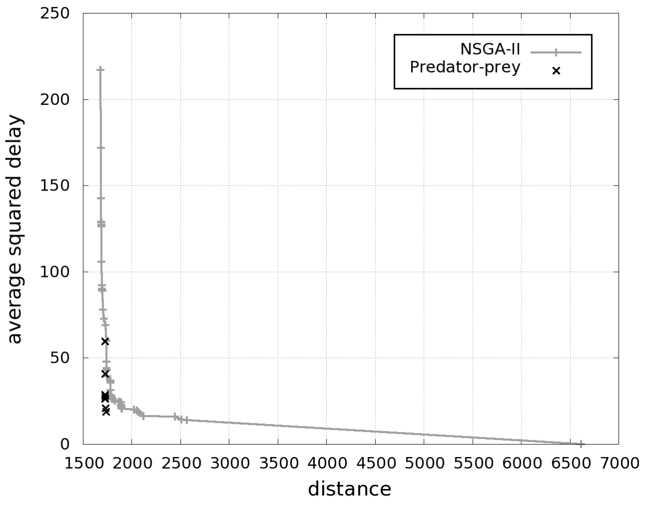

Fig. 13. NSGA-II vs. PP: RC101 average squared delay

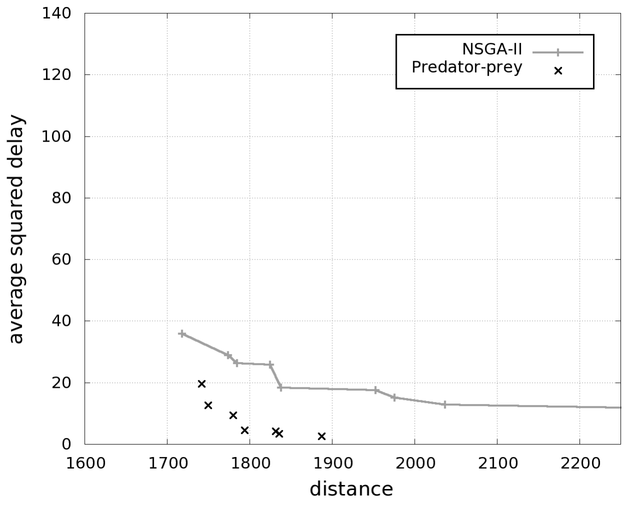

Fig. 14. NSGA-II vs. PP after 60 seconds

TABLE I
PREDATOR-PREY VS. SINGLE-OBJECTIVE ALGORITHM: AVERAGE DISTANCE VS. AVERAGE ROBUSTNESS FOR AVERAGE DELAYS

	Predator-Prey		Single-obj.		Deviation	
	Avg.D.	Avg.R.	Avg.D.	Avg.R.	Avg.D.	Avg.R.
C1	849.59	0.24	831.42	8.40	+2.2%	-97.1%
C2	607.52	0.11	590.02	41.22	+3.0%	-99.7%
R1	1241.98	2.75	1205.18	32.40	+3.1%	-91.5%
R2	888.06	0.45	886.81	1.69	+0.1%	-73.4%
RC1	1435.87	3.91	1404.96	39.87	+2.2%	-90.2%
RC2	1024.59	0.41	1018.15	2.67	+0.6%	-84.7%
Total					**+1.9%**	**-93.4%**

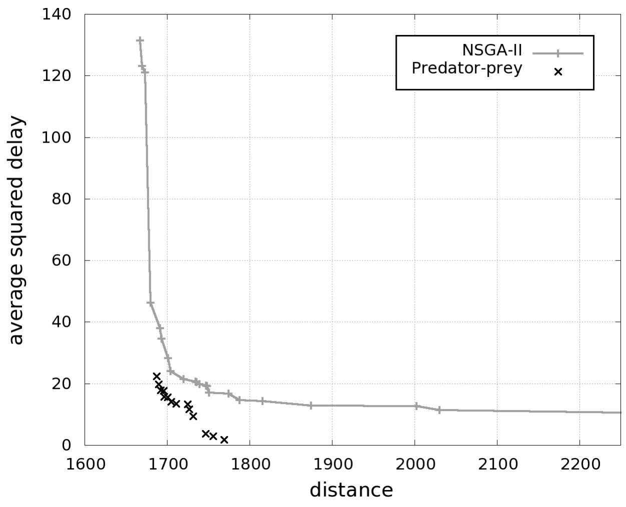

Fig. 15. NSGA-II vs. PP after 180 seconds

TABLE II
PREDATOR-PREY VS. SINGLE-OBJECTIVE ALGORITHM: AVERAGE DISTANCE VS. AVERAGE ROBUSTNESS FOR AVERAGE SQUARED DELAYS

	Predator-Prey		Single-obj.		Deviation	
	Avg.D.	Avg.R.	Avg.D.	Avg.R.	Avg.D.	Avg.R.
C1	849.59	1.90	831.42	46.66	+2.2%	-95.9%
C2	607.52	0.45	590.02	832.86	+3.0%	-99.9%
R1	1241.85	12.45	1205.18	217.78	+3.0%	-94.3%
R2	888.06	1.21	886.81	7.44	+0.1%	-83.8%
RC1	1434.73	29.58	1404.96	328.76	+2.1%	-91.0%
RC2	1024.37	1.70	1018.15	12.07	+0.6%	-85.9%
Total					**+1.9%**	**-96.5%**

The numbers show that for both robustness metrics the average robustness could be improved significantly for just minor additions in average distance. The exceptions here are the R2 and RC2 instances where the gain in robustness is insignificant, which can be explained by the high degree in robustness already existing in the standard solutions, which again is a result of the very wide time-windows.

C. *Runtime Analysis*

To compare solution quality over runtime the results have been tracked over time for the R101 instance and plotted into the same diagrams after 60, 180, 300, 900 and 1500 seconds (figures 14, 15, 16, 17 and 18), the predator prey algorithm stopped after 300 seconds.

It can be seen that the predator prey algorithm produces more robust solutions much faster. The general better performance of the predator prey can be explained by the decreased number of fitness-function evaluations.

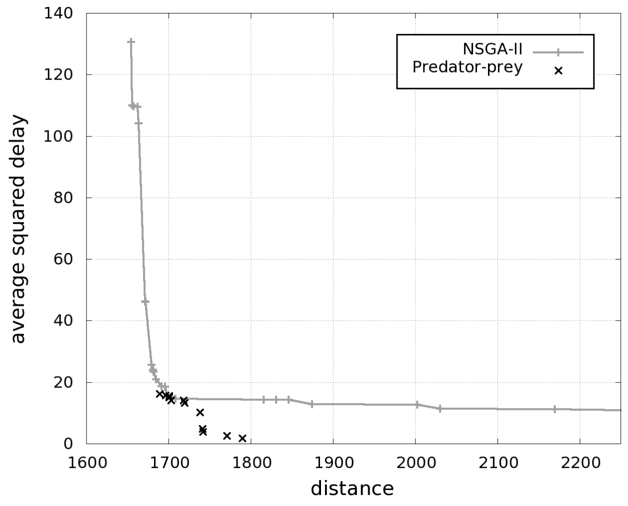

Fig. 16. NSGA-II vs. PP after 300 seconds

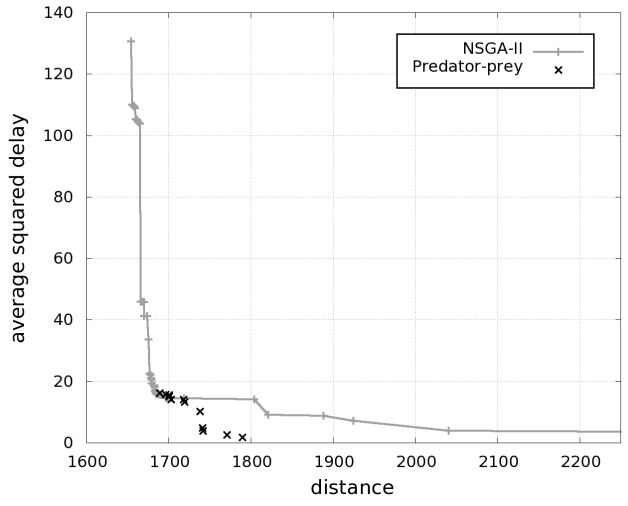

Fig. 17. NSGA-II vs. PP after 900 seconds

The property of creating more robust solutions without a concomitant increase in distance, can be explained by the way individuals are created by the variation and selection operators. In the NSGA2 the mutation leads to low distance solutions as the insertion-rules are based on the default driving times and robustness is just a side-effect exploited by the evolutionary selection. In the predator prey algorithm the insertions inside of the mutation operator is based on the modified travel-times which leads to low distance and robust solutions more quickly. As a consequence, it could be argued that the Homberger/Gehring mutation is not an expedient operator for the NSGA2 – still with this many fitness-function evaluations a smart mutation operator is needed to assist fast convergence to the short distance solutions, especially as distance can (in this case) be assumed to be of dominating importance to robustness.

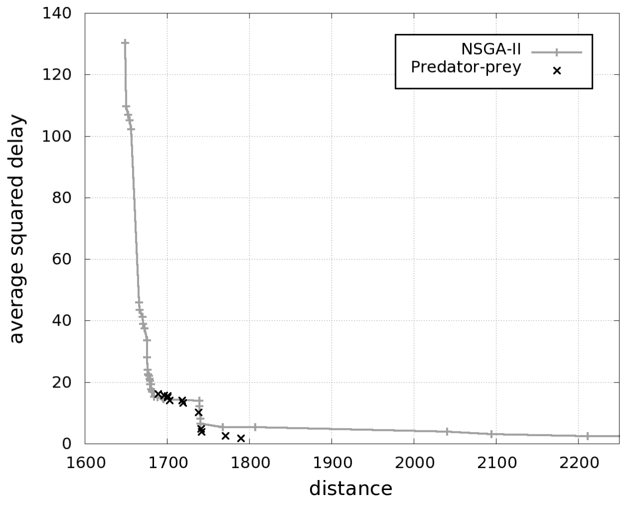

Fig. 18. NSGA-II vs. PP after 1500 seconds

VI. Conclusion and Outlook

A. Conclusion

The main insight in this work is, that in vehicle routing problems it is possible to increase the robustness of a solution significantly by accepting just minor detours. The methods presented are able to model the situations or circumstances, in which solutions are subject to uncertainties, effectively. Still, both algorithms have runtimes which will not be accepted by most real world planners – especially not in vehicle routing. The basic approach can be transferred to other problems in which runtime is not an issue, but it can generally be concluded that the runtime is too high.

B. Outlook

While, due to the high runtime, the approach is probably not directly applicable, results may help to evaluate simpler but faster algorithms. Straighforward ideas, like inserting buffers into critical spots, might create results that can be evaluated by the proposed scenarios and plotted into the same diagrams as the NSGA2. The results of fast algorithms can then be compared to the full range of reachable robust solutions.

We will next apply these approaches to problems which are more closely based on scenarios, like site location problems, which also have less demanding runtime requirements. Another application might also be to address flexibility – scenarios might represent situations in which more customers need to be integrated into the plan, and flexibility could be defined as the cost for integrating them.

VII. Acknowledgment

This work is part of the project RoKoKo and was supported in part by the Bundesministerium für Bildung und Forschung (BMBF) under grant no. FKZ 01IS08012D.

References

[1] Toth, P. and Vigo, D. *The Vehicle Routing Problem*, Society for Industrial & Applied Mathematics, 2002.

[2] Sörensen, K. and Sevaux, M., *Robust and Flexible Vehicle Routing in Practical Situations*, Proceedings of the 5th Triennial Symposium on Transportation Analysis, 2004

[3] Cordeau, J.F. and Gendreau, M. and Laporte, G. and Potvin, J.Y. and Semet, F., *A Guide to Vehicle Routing Heuristics*, Journal of the Operational Research Society, Volume 5, pp.512–522, 2002.

[4] Laporte, G. and Louveaux, F. and Mercure, H., *The Vehicle Routing Problem with Stochastic Travel Times*, Transportation Science, Volume 26, pp.161–170, 1992.

[5] Lambert, V. and Laporte, G. and Louveaux, F., *Designing Collection Routes through Bank Branches*, Computers and Operations Research, Volume 20, pp.783–791, 1993.

[6] Kenyon, AS and Morton, DP, *Designing Collection Routes through Bank Branches*, Transportation Science, Volume 37, pp.69–82, 2003.

[7] Russell, RA and Urban, TL, *Vehicle Routing with Soft Time Windows and Erlang Travel Times*, Journal of the Operational Research Society, Volume 59, pp.1220–1228, 2008.

[8] Scholl, A., *Robuste Planung und Optimierung*, Physica-Verlag, Heidelberg, 2001.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, Volume 6, pp.182–197, 2002.

[10] Laumanns, M. and Rudolph, G. and Schwefel, H.P., *A Spatial Predator-Prey Approach to Multi-objective Optimization: A Preliminary Study*, Proceedings of the 5th International Conference on Parallel Problem Solving from Nature-PPSN V, pp.241–249, 1998.

[11] Coello, C.A.C. and Van Veldhuizen, D.A. and Lamont, G.B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2002.

[12] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold New York, 1991.

[13] Homberger, J. and Gehring, H., *Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows*, Infor, Volume 37, pp.297–318, 1999.

[14] Solomon, M.M., *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints*, Operations Research, Volume 35, pp.254–265, 1987.

[15] Tan, K.C., Cheong, C.Y. and Goh, C.K., *Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation*, European Journal of Operational Research, vol. 177, pp. 813-839, 2007.