MULTI-OBJECTIVE EVOLUTIONARY SIMULATION-OPTIMIZATION OF PERSONNEL SCHEDULING

Anna Syberfeldt[1], Martin Andersson[1], Amos Ng[1], and Victor Bengtsson[2]

[1]Virtual Systems Research Center, University of Skövde, Skövde, Sweden
[2]PostNord AB, Solna, Sweden

ABSTRACT

This paper presents an evolutionary multi-objective simulation-optimization system for personnel scheduling. The system is developed for the Swedish postal services and aims at finding personnel schedules that minimizes both total man hours and the administrative burden of the person responsible for handling schedules. For the optimization, the multi-objective evolutionary algorithm NSGA-II is implemented. In order to make the optimization fast enough, a two-level parallelisation model is being adopted. The simulation-optimization system is evaluated on a real-world test case and results from the evaluation shows that the algorithm is successful in optimizing the problem.

KEYWORDS

Multi-objective evolutionary optimization, NSGA-II, hill climbing, personnel scheduling, case study.

1. INTRODUCTION

The use of a simulation in combination with an optimization algorithm, so called simulation-optimization, is a commonly used technique in industry today to find the best parameter values for a system or a process. While traditional optimisation methods have been unable to cope with the complexities of many real-world problems approached by simulation, evolutionary algorithms have proven to be highly useful in dealing with complex problems [1-2]. Such problems include personnel scheduling [3-4], which is of focus for this study. The proven success of evolutionary algorithms has motived to use the approach for tackling the simulation-optimization problem at hand in this study, namely scheduling of personnel within one of the largest corporations in Sweden; PostNord. PostNord is the Swedish postal services with more than 30 000 employees and a revenue of approximately 25 billion SEK. Core business comprises distribution of post/mail and logistics, and PostNord is one of the largest actors in these areas in the Nordic region. As the Nordic postal market is fully deregulated, mail business is a highly competitive market. New operators are becoming established, especially in larger cities, and can compete with lower fees as they are not covered by the same regulations as PostNord. Facing national and international actors operating in the same business areas puts high demands on efficient mail operations, and additional pressure arises from the legal directives that PostNord is obligated to follow, specifying that mail operations must be fast, reliable, and cost-efficient. These challenges are not unique for the Swedish postal market, but postal administrations all over the world face the same requirement to continuously analyze and improve their services [5].

This paper describes a study aiming at improving PostNord's processes by developing a simulation-optimization system for personnel scheduling. The current process for creating schedules is manual, and creating efficient schedules with low personnel overhead is difficult. The aim of this project is to replace major parts of this manual process with an automatic simulation-optimization system. The system is not meant to totally replace the human expert, but rather to act as a decision support tool. The reason for not replacing the human is that there are many small, but complicated, details that will be hard to implement into a system.

In the next chapter, the optimization problem to be solved is described in further detail. In Chapter 3, the design and implementation of the optimization algorithm used to tackle the problem is presented. Chapter 4 describes the simulation-optimization procedure and how this is made more efficient through parallelisation. In Chapter 5, results from applying the evolutionary simulation-optimization on a real-world personnel scheduling problem are described. Chapter 6, finally, presents conclusions from the study and outlines interesting future work.

2. OPTIMIZATION PROBLEM

2.1. Overview

PostNord is mandated to make postal services available all over Sweden and handles approximately 20 million mail per work day. The governmental mandate states that mail must be collected and delivered on every workday and at least five days a week nationwide. Furthermore, PostNord must guarantee that at least 85 percent of priority mail posted before a specific time must be delivered during the following workday – wherever it is addressed to in Sweden. The mail distribution process basically consists of the following operations (Figure 1):

1. *Collection (3 - 8 pm)*
   Mail are collected from about 30 000 collection boxes and 3 000 retail service outlets during the afternoon and brought to the closest mail processing facility.

2. *Input sorting (8 - 11 pm)*
   At the 11 mail sorting facilities, all mail are sorted and coded with a barcode according to the destination mail processing facility.

3. *Inter-regional transportation (11 pm - 3 am)*
   Overnight airplanes, freight trains, and trucks move the mail from the origin mail processing facility to the destination mail processing facility.

4. *Output sorting (3 - 5 am)*
   All mail are sorted again, this time to determine their regional mail carrier centre.

5. *Intra-regional transportation (5 - 9 am)*
   The mail are transported to the regional mail carrier centre (there are about 600 regional mail carrier centers) where a postman sorts them according to the delivery route.

6. *Distribution (9 am – 2 pm)*
   About 15 000 mail carriers deliver the mail to the final addresses.

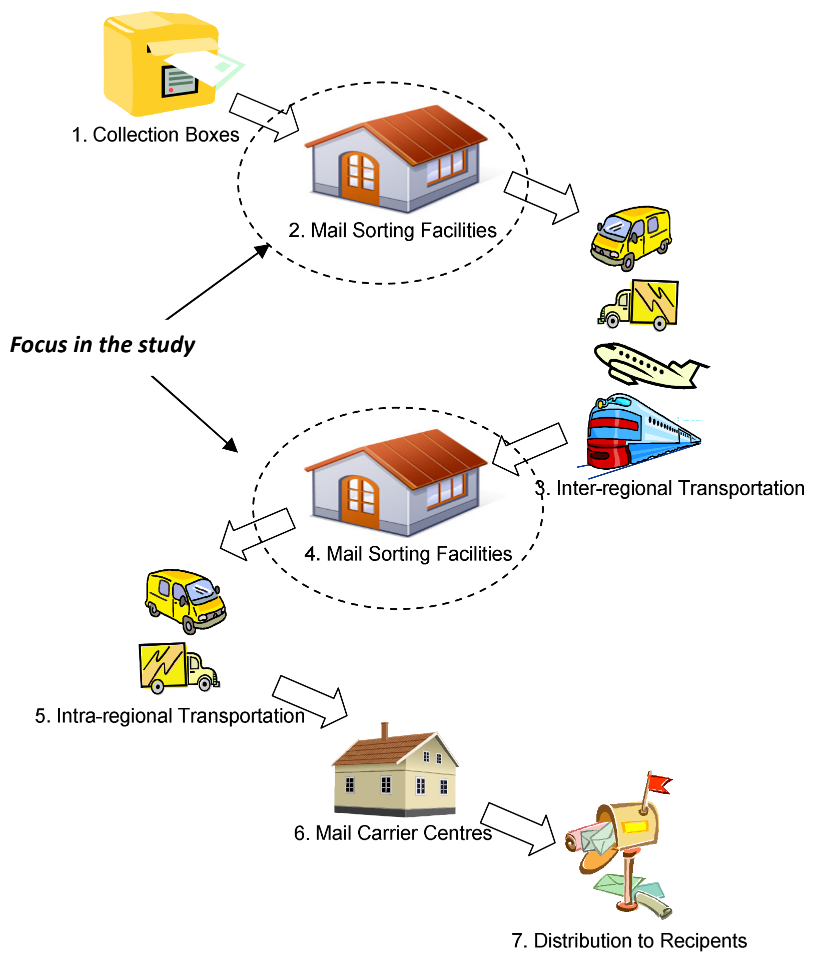
1. Collection Boxes
2. Mail Sorting Facilities
Focus in the study
3. Inter-regional Transportation
4. Mail Sorting Facilities
5. Intra-regional Transportation
6. Mail Carrier Centres
7. Distribution to Recipents

Figure 1: Mail distribution process.

The focus of this study is on step 2 and step 4 of the distribution process, namely the mail sorting. A typical sorting facility has about 150 employees working in three shifts, and the aim of the study is to optimize the scheduling of the personnel. The next subchapter continues by describing the optimization problem into further detail.

2.2. Problem specification

The overall task of the optimization procedure is to take a given personnel demand for a given time period and create a set of personnel schedules with the number of people needed for each schedule, so that they together satisfy the given personnel demand while also fulfilling government laws, union regulations and company policies.

Two objectives need to be considered in the optimization:

a) Minimize the total amount of scheduled man hours (in order to minimize the cost), and
b) To minimize the total amount of schedules (in order to reduce the administrative burden of handling a large amount of schedules).

The two objectives are usually conflicting with each other, because the more schedules that is used the easier it is to fit the schedules to the personnel demand. In the next chapter, it is described how this conflict is handled in the optimization algorithm.

3. OPTIMIZATION ALGORITHM

This chapter describes the multi-objective method used in the optimization, as well as the optimization algorithm and its parameters.

3.1. Pareto optimization

The difficulty with conflicting optimization objectives is that there is usually no single optimal solution with respect to both objectives. Instead of a single optimum, there is a set of optimal trade-offs between the conflicting objectives, called Pareto-optimal solutions [6]. Figure 2 illustrates the Pareto concept for a minimization problem with two objectives f_1 and f_2. In this example, solutions A-D are non-dominated, i.e., Pareto optimal, since for each of these solutions no other solution exists that is superior in one objective without being worse in another objective. Solution E is dominated by B and C (but not by A or D, since E is better than these two in f_1 and f_2, respectively). Different Pareto ranks can also be identified among solutions. Rank 1 includes the Pareto-optimal solutions in the complete population, and rank 2 the Pareto-optimal solutions identified when temporarily discarding all solutions of rank 1, and so on.

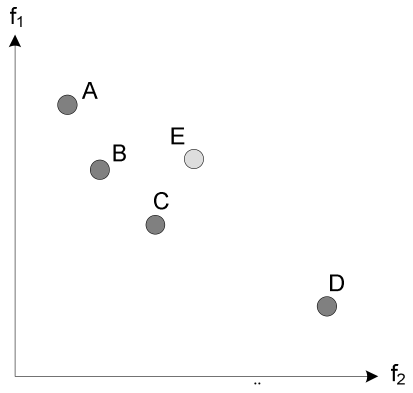

Figure 2. Illustration of dominance.

In order to manage multiple objectives, specific optimization algorithms have been suggested. Instead of only seeking a single optimum, these algorithms maintain a set of Pareto-optimal solutions. One of the most efficient algorithms for Pareto optimization is the elitist non-dominated sorting genetic algorithm (NSGA-II) [7]. Due to the algorithm's proven efficiency, this is the optimization algorithm selected for the study. The details of the algorithm are described in the next subchapter.

3.2. NSGA-II algorithm

NSGA-II is basically a genetic algorithm extended with features for handling multiple trade-off solutions [8]. Like an ordinary genetic algorithm, NSGA-II maintains a population of solutions (in this case schedules) and refines these solutions through generations. In the refinement process, crossover and mutation operators are used to create offspring solutions that become part of the next generation. The basic steps involved in evolution algorithms are presented below (a complete description of genetic algorithms can be found in [9]):

```
/* Start of algorithm */
Initialize population
Evaluate the fitness of solutions in the population
repeat
  Select solutions to reproduce
  Form a new generation of population through
    crossover and mutation
  Evaluate the new solutions
until terminating condition
/* End of algorithm */
```

Contrary to an ordinary genetic algorithm, NSGA-II does not seek a single optimum, but maintains a set of Pareto-optimal solutions. The selection of solutions to continue to the next generation of the population is done from a set R, which is the union of a parent population and an offspring population (both of size N) [7]. Non-dominated sorting is applied to R to identify Pareto fronts, and the next generation of the population is formed by selecting solutions from one of the fronts at a time. The selection starts with solutions in the best Pareto front, then continues with solutions in the second best front, and so on, until N solutions have been selected. If there are more solutions in the last front than there are remaining to be selected, niching is applied to determine which solutions should be chosen. In other words, the highest ranked solutions located in the least crowded areas are the ones chosen. All the remaining solutions are discarded. The selection procedure is illustrated in Figure 3 (adopted from [6]).

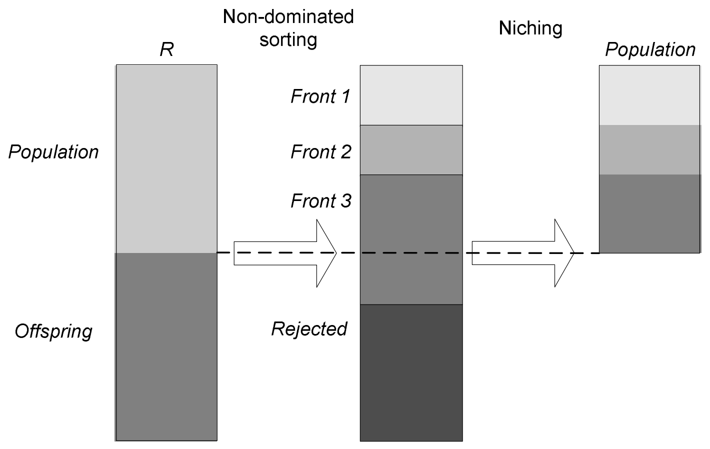

Figure 3. Selection procedure in NSGA-II.

Two vital parts in NSGA-II, as in all evolutionary algorithms, are the crossover and mutation operators. The implementation of these operators is further described in the next two subchapters.

3.4. Crossover operator

The crossover operator is used to exchange schedules between solutions. It takes two solutions as input (the parents) and produces two solutions as output (the children). To select the schedules to pick from the first parent the crossover operator adds all schedule indices into a list, shuffles it and then it randomly selects a subset of that list to insert into the first child, the remaining schedules are added to the second child. Then the same procedure is performed for the second parent. The end result is two children that are a mix of the schedules from the two parents.

3.5. Mutation operators

Seventeen different mutation operators have been implemented in the optimization algorithm, which are described in the list below. The operators differ from the crossover in that they make smaller changes, usually only affecting one schedule.

- Add one schedule, based on shifts from other schedules.
- Add one schedule, from randomly generated shifts.
- Remove one schedule.
- Add one shift to a schedule.
- Remove one shift from a schedule.
- Shorten one shift.
- Extend one shift.
- Pick a schedule and create a new schedule for each shift.
- Pick a schedule and create a new schedule from a subset of all shifts.
- Merge schedule, pick a schedule and merge a subset of all other schedules into that one. Remove the schedules that are completely merged.
- Move one shift backwards, cannot move beyond the preceding shift.
- Move one shift forwards, cannot move beyond the succeeding shift.
- Move one shift, not limited by other shifts.
- Move one shift from one schedule to another.
- Move the day rest.
- Move the week rest.
- Rearrange the breaks on one shift.

In the next chapter, the implementation of the optimization algorithm in a simulation-optimization procedure is described.

4. SIMULATION-OPTIMIZATION

This chapter describes the simulation model used for evaluating schedules generated by the optimization algorithm. It also describes the iterative simulation-optimization process and how this has been made more efficient through parallelisation.

4.1. Simulation model

For evaluating schedules generated by the optimization algorithm, a simulation is used. The simulation is implemented using C++ based on the software library Cbc (https://projects.coin-or.org/Cbc) which is an open-source mixed integer programming solver. The task of the linear programming solver is to assign people to schedules. This is an integer problem, since it's not possible to assign, for example, 4.6 people to a schedule. The issue is that solving linear integer problems is computationally expensive. One method of mitigating this is to relax the integer problem (and round solution values up), thereby sacrificing solution quality for solution speed. This is the approach taken in this paper, since the relaxed problem is at least an order of magnitude faster to solve, easily overcoming the loss of solution quality.

4.2. Simulation-optimization procedure

From an overall perspective, the simulation takes a set of schedules as input, finds the optimal assignment of people to these schedules given a specific demand (using linear programming) and eventually returns the total amount of man hours and feasibility (if the schedule is valid, i.e. fulfils basic constraints). Based on the results returned, the optimization algorithm generates a new set of schedules and sends these to the simulation model. This process of generating and evaluating schedules (shown in Figure 4) then continues until the user-defined stopping criterion is met.

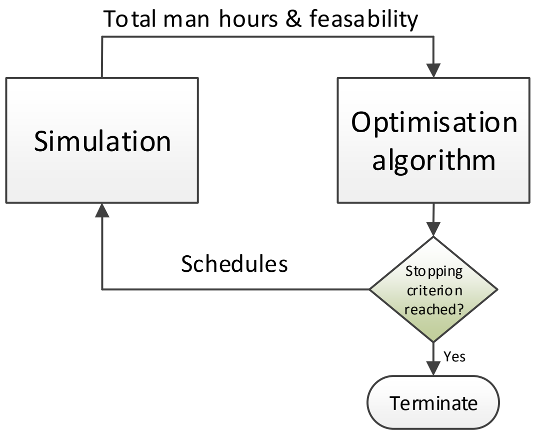

Figure 4. Iterative simulation-optimization procedure.

For the optimization to find sufficiently good solutions, a large amount of simulations is needed. Although the simulation is fast (one simulation run takes approximately 30 milliseconds), the time consumption of the optimization process becomes quite large due to the high complexity of the problem. To reduce the time consumption, a parallel approach with multiple simulation agents is adopted. This approach is further described in the next subchapter.

4.3. Parallelisation

For increased efficiency, a number of parallel computing resources are used in the simulation-optimization. Contrary to many other systems, not only the simulation but also the optimization algorithm is distributed and run in parallel. This is done because when the simulation is fast and/or the amount of available computing resources is large, there comes a point where the optimization algorithm can't generate solutions fast enough to keep the computing resources completely busy. To scale the system beyond this point the algorithm must be able to run in parallel instances. Parallel instances of an algorithm that don't share solutions are equivalent to running multiple serial replications of the same algorithm. This can be useful to speed up a series of replications, but it won't increase the speed of single run. In order to do that, solutions must be shared between the algorithms so they can benefit from each other's work.

There are three major parallelization strategies for metaheuristic algorithms, which fall into three distinct hierarchical levels, see Figure 5 [10]. On the algorithmic level multiple independent or cooperating algorithms are used to achieve parallelization. This is problem independent and it also has the potential of affecting the behavior of the algorithm. The iteration level concerns the parallelization of a single algorithm. An example of this would be in a population based algorithm the evaluation of solutions could be performed in parallel. This is also problem independent but it does not affect the behavior. The last level is the solution level, which concerns the parallelization of one single solution evaluation. Either the objective function is divided into several parts that can be executed in parallel or the input data is partitioned. This is problem dependent since both these approaches are tightly coupled to the problem at hand.

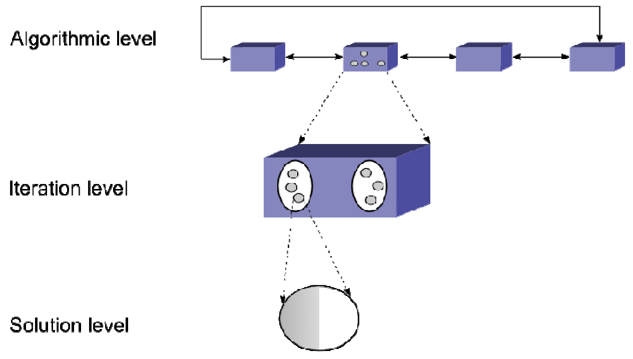

Figure 5. Parallelization levels.

In this system the parallelization is both on the algorithmic level (creating solutions in parallel) and on the iteration level (evaluating solutions in parallel). There is no parallelization on the solution level. By having the parallelization on both the algorithmic and iteration levels it's possible to achieve a system that can adapt to most problem instances (problems differ in the time they take to create and evaluate solutions), which allows for high utilization of computing resources by avoiding bottlenecks in either the schedule generation or assigning people to schedules tasks.

When an optimization instance finds a better solution it is always directly sent to all other currently running optimization instances. This means that there's only a small time window where the instances differ in what solutions they have. The advantage is that each optimization instances is always operating on the global best solutions (or something very close to it), with the disadvantage that there are a lot of traffic between the optimization instances in the beginning when it is easy to find better solutions. That disadvantage is mitigated by the fact that it is possible to run multiple optimization instances on the same computer, which communicate by message passing via system memory in order to avoid excessive network traffic.

The next chapter continues by describing the evaluation of the simulation-optimization system.

5. EVALUATION

This chapter describes the real-world test case used to evaluate the simulation-optimization system and presents results from the evaluation.

5.1. Test case

A real-world test case is being used for the evaluation. The goals of the test case are defined as:

- Make sure that the system generates schedules that obey laws and regulations.
- Find the optimal parameter settings to improve the effectiveness of the system.
- Evaluate the quality of the schedules generated by the system.
- Find the requirements of the graphical user interface.

For the test case, one of the departments at PostNords mail hub in Gothenburg was chosen. The hub has about 150 employees working in three shifts. As input to the test case, the personnel demand for one month was collected with a time resolution of half an hour, together with actual schedules that was used at the time.

5.2. Baseline Comparison

For assessment of the performance of the NSGA-II optimization algorithm, an additional optimization algorithm is implemented for comparison. The additional algorithm implemented is hill climbing, which is selected since it is well-known, simple to implement and often used for baseline comparisons. Hill climbing is an iterative algorithm that belongs to the family of local search algorithms [11]. The algorithm starts with a random solution to the problem and attempts to find a better solution by mutating (changing) the solution. If the mutation produces a better solution, it is kept and the procedure is repeated until no further improvements can be made.
The hill climbing is implemented with the same mutation operators as the NSGA-II algorithm and also allocated the same number of total simulation evaluations. For increased efficiency, the hill climbing algorithm is implemented to constrain the amount of schedules each solution can have to a predetermined amount. This is done because even though increasing the amount schedules will generally allow for better solutions, there exist a point where this is no longer true. Further, this point is usually well beyond what is considered a reasonable amount of schedules. This fact is utilized by the hill climbing, which allows it to store every solution with a unique amount of schedules and then individually improve these solutions, using the second objective which is the total amount of scheduled time. When a mutation operation cause a solution to gain or lose schedules, this creates a new solution which can then be compared to the current best with that specific amount of schedules.

5.3. Results

The optimization results achieved by the NSGA-II algorithm and the hill climbing are presented in Table 1. As shown in the table, the results indicate that NSGA-II outperforms the hill climbing algorithm.

Table 1. Results (best solution found).

Algorithm	**Total man hours** (to be minimized)	**No. of schedules** (to be minimized)
NSGA-II	20 676	31
Hill climbing	24 251	41

The results from the optimization are also shown in Figure 6. In the figure, the y-axis represents the total amount of man hour (as this is considered the most interesting of the two objectives) and the x-axis represents number of evaluations. As can be seen in the figure, the progress of the two algorithms differs mostly in the beginning of the search and is quite similar towards the end.

It is worth to notice that from an algorithmic perspective the hill climbing algorithm is considerable less time consuming than the NSGA-II. While the NSGA-II algorithm took 79 minutes to run in the real-world test case, the hill climbing algorithm took only 32 minutes (with the exact same number of simulation evaluations allocated for both algorithms). The reason that hill climbing is more efficient from an algorithmic perspective is because it avoids computationally expensive operations like non-dominated sorting and crowding distance calculations that is used in the NSGA-II algorithm.

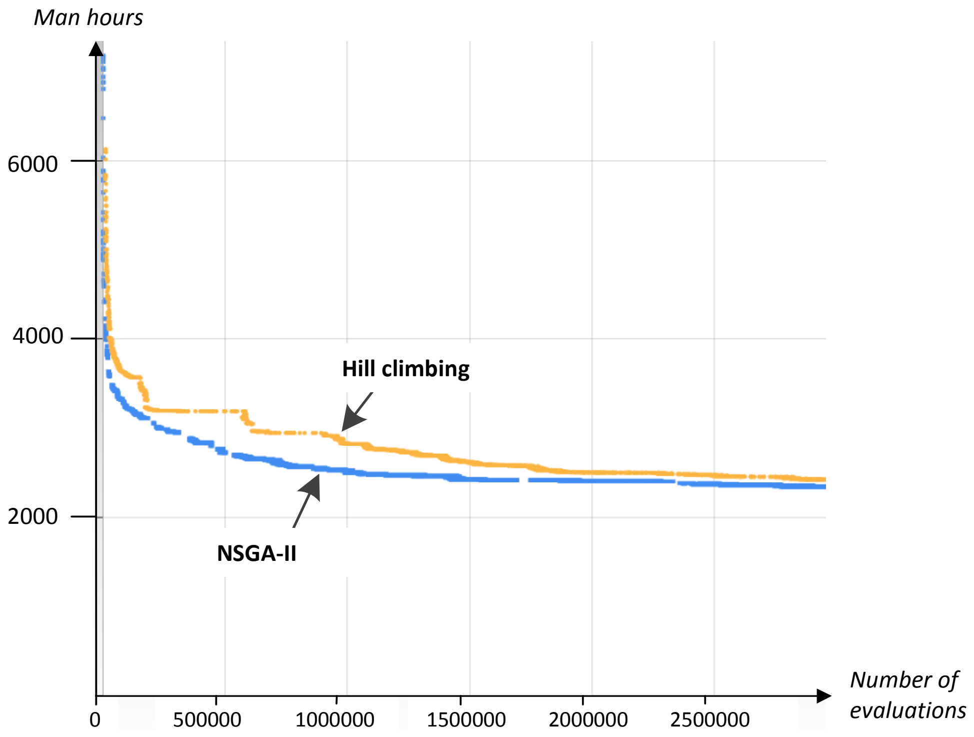

Figure 6. Results over time

PostNord has evaluated the schedules obtained by the simulation-optimization system and they see an advantage in that the system heavily reduces the human effort associated with creating schedules. Compared to the manual approach of creating schedules used at PostNord so far, a lot of time and effort can be saved for the person responsible for creating schedules. Furthermore, it has been proved that the optimization finds more efficient personnel schedules than the human scheduler does. The exact amount of money saved by using the optimization has not yet been investigated, but considering that the average salary of an employee is approximately 2700 euro per month and there are about 150 people working at each sorting terminal, every percentage of increased efficiency obtained by the optimization means large savings. Another advantage with the optimization is that it makes it easy to obtain schedules with certain focuses, for example, a minimum number of full-time schedules, as the decision makers' preferences are considered in the optimization process.

6. CONCLUSIONS

This paper presented a simulation-optimization system for improving personnel schedules in the Swedish postal services. The optimization aims at finding schedules that minimizes both the total man hours and the administrative burden of the person responsible for handling schedules by reducing the number of schedules. The two objectives of minimizing man hours and number of schedules are conflicting with each other, because the more schedules the easier it is to fit the schedules to the personnel demand. The difficulty with conflicting optimization objectives is that there is usually no single optimal solution with respect to both objectives, and to handle this difficulty a specialized multi-objective algorithm called NSGA-II is implemented. For evaluating solutions, the NSGA-II algorithm uses a simulation based on the software library Cbc. For increased performance of the simulation-optimization, parallelization is implemented both on the algorithmic level (creating solutions in parallel) and on the iteration level (evaluating solutions in parallel). This allows for high utilization of computing resources and quick response time from the simulation-optimization.

The simulation-optimization system is evaluated on a real-world test case obtained from one of PostNord's mail hubs in Sweden having about 150 employees. The schedules being the outcome of the test case have been evaluated by the company and the results look very promising. These results have so far only been verified theoretically, but in the future optimized schedules can hopefully be implemented and evaluated also in reality.

A potential improvement of the system is to transform it from locally installed software into web-based software. Being web-based, the system would accessible from anywhere with an internet connection, and not only from the specific computer that has the system installed. The maintenance of web-based systems is also easier, since updates can be made on the server and reaches the clients immediately. Furthermore, with a web-based system it is possible to easily share schedules in real-time among stakeholders.

Another improvement of the system would be to programmatically integrate it with the personnel handling systems used at PostNord. Such integration would eliminate the need of manually providing input regarding personal demand, available personnel, etc. before an optimization is started. With integration, the optimization can be fully automatic and save even more human efforts.

ACKNOWLEDGEMENTS

This work has been carried out within the BLIXTSIM project which is partially financed by the Knowledge Foundation (KK Stiftelsen), Sweden. The authors gratefully acknowledge the Knowledge Foundation for the provision of research funding.

REFERENCES

[1] Coello, C. and Lamont, G. (2004) "Applications of Multi-Objective Evolutionary Algorithms", Advances in Natural Computation, World Scientific Publishing, Singapore.
[2] Emmerich, M. (2005) "Single- and Multi-Objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels", PhD thesis, University of Dortmund.
[3] Aickelin, U., Burke, E.K. and Jingpeng, L. (2008) "An Evolutionary Squeaky Wheel Optimization Approach to Personnel Scheduling". IEEE Transactions on Evolutionary Computation, Vol.13, No. 2, pp 433-443.
[4] Volker Nissen, Maik Günther (2009) "Staff Scheduling with Particle Swarm Optimisation and Evolution Strategies". Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, Vol. 5482, pp 228-239.
[5] Larsen, N.E. 2003. "Simulation – a key tool to accelerate and add confidence to postal network configuration". Proceedings of the 2003 Winter Simulation Conference, pp 1585-1592, Orlando, Florida.
[6] Deb, K. 2004. "Multi-objective optimization using evolutionary algorithms", Chichester: John Wiley & Sons, Ltd.
[7] Deb, K., Pratap, A., Agarwal, S. andMeyarivan, T. (2000) "A fast and elitist multi-objective genetic algorithm NSGA-II", KanGAL Report 2000001, Indian Institute of Technology Kanpur, India.
[8] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002) "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 2, 182-197.
[9] Bäck, T., Fogel, D. and Michalewicz, Z. (eds) 1997. "Handbook of Evolutionary Computation". Oxford University Press.
[10] Talbi, E.-G. (2009) "Metaheuristics: From Design to Implementation", John Wiley & Sons, pp 478-504.

[11] Russell, S.J. and Norvig, P. (2003. “Artificial Intelligence: A Modern Approach”. 2nd ed, Upper Saddle River, New Jersey: Prentice Hall, pp 111–114.

Authors

ANNA SYBERFELDT is an associated professor at the University of Skövde, Sweden. She holds a Ph.D in Computer Science from the De Montfort University, UK and a Master’s degree in Computer Science from the University of Skövde, Sweden. Her research interests include simulation-based optimization, artificial intelligence, metaheuristics, and advanced information technology with applications in logistics and manufacturing.

MARTIN ANDERSSON is a Ph.D candidate at the University of Skövde, Sweden. He has a B.Sc. in Computer Science and a M.Sc. in Automation Engineering from the University of Skövde. His research interests include simulation-based optimization and parallel metaheuristics.

AMOS NG is a Professor at the University of Skövde, Sweden. He holds a Ph.D. degree in Computing Sciences and Engineering. His research interests lies in applying multiobjective optimization for production systems design & analysis.

VICTOR BENGTSSON is an Optimization- & Simulation Specialist at PostNord AB. He holds a Master degree in mathematics from the University of Linköping, Sweden. His research interests include production simulation, simulation-based optimization, multi-objective optimization, set covering optimization.