Memorandum 1982 (May 2012). ISSN 1874-4850. Available from: http://www.math.utwente.nl/publications
Department of Applied Mathematics, University of Twente, Enschede, The Netherlands

Optimal staffing under an annualized hours regime using Cross-Entropy optimization

Egbert van der Veen*,[1,2,3], Richard J. Boucherie[2,3], and Jan-Kees C.W. van Ommeren[3]

[1]ORTEC, The Netherlands
[2]Center for Healthcare Operations, Improvement, and Research (CHOIR), University of Twente, The Netherlands
[3]Stochastic Operations Research, Department of Applied Mathematics, University of Twente, The Netherlands

May 29, 2012

Abstract

This paper discusses staffing under annualized hours. Staffing is the selection of the most cost-efficient workforce to cover workforce demand. Annualized hours measure working time per year instead of per week, relaxing the restriction for employees to work the same number of hours every week,

To solve the underlying combinatorial optimization problem this paper develops a Cross-Entropy optimization implementation that includes a penalty function and a repair function to guarantee feasible solutions. Our experimental results show Cross-Entropy optimization is efficient across a broad range of instances, where real-life sized instances are solved in seconds, which significantly outperforms an MILP formulation solved with Cplex. In addition, the solution quality of Cross-Entropy closely approaches the optimal solutions obtained by Cplex. Our Cross-Entropy implementation offers an outstanding method for real-time decision making, for example in response to unexpected staff illnesses, and scenario analysis.

Keywords: Combinatorial Optimization, Cross-Entropy optimization, annualized hours, personnel staffing, metaheuristics, knapsack problem

1 Introduction

Businesses and institutions frequently have to make decisions on the contract-mix and skill-mix of their workforce. In addition, some organizations have

*corresponding author: Egbert.vanderVeen@ortec.com

adapted the annualized hours regime (see, e.g., [24, 30]), which allows them to measure working time per year, rather than per week or per month. This extra flexibility implies decision making on the distribution of workforce capacity throughout the year. This is further complicated by both planned absences (e.g., planned holidays and education) and expected, but unknown, absences (e.g., illnesses). Moreover, in many settings demand for skilled workers fluctuates throughout the year. Although workforce demand is uncertain to some extent, we consider a deterministic variant, and assume that operational demand deviations can be captured by letting employees work extra or hiring subcontractors. Our main challenge is to distribute workforce capacity such that demand for skilled workers is met as cost-efficiently as possible, and unnecessary hiring of subcontractors is minimized.

The problem studied here considers two aspects that are in literature often addressed separately. On the one hand we have the 'staffing problem', which determines the 'optimal' workforce, and on the other hand we have the 'annualized hours' problem, which is about distributing the workforce capacity optimally over the year. We investigate how Cross-Entropy (CE) optimization can be used to solve a problem that considers both staffing and annualized hours. A problem we refer to as the staffing under annualized hours (SUAH) problem. CE is widely used for rare event simulation, but is also used to solve combinatorial optimization problems. In essence, CE iteratively generates samples of solutions. Based on the best solutions in a sample, the sampling distribution's parameters are chosen to minimize the Cross-Entropy 'distance' between a sampling distribution with these sampling parameters, and the current sampling distribution. With these updated parameters CE aims to generate better solutions in the next iteration.

We apply CE optimization, which is successfully applied to problems related to SUAH, like the multidimensional knapsack problem (see, e.g., [22]) and the capacitated facility location problem (see, e.g., [7]). The objectives of multidimensional knapsack and capacitated facility location are to select an 'optimal' set of items and an 'optimal' set of (supply) locations respectively. In [7] the capacitated facility location problem is solved using CE and in [9, 19, 36] CE is used to solve (variants of) knapsack problems. In addition to successful CE applications to related problems, metaheuristics, like tabu search and simulated annealing, have also shown to work well on knapsack problems [22]. An overview of related literature is found in Section 2.

The results of our CE implementation show that, across a broad range of instances, CE considerably outperforms an MILP implementation solved with Cplex. Specifically instances with 10 or more planning periods, the periods over which the annualized hours problem distributes available capacity, are solved relatively quickly by CE. In practice, the planning horizon often contains dozens of planning periods, for example, if the planning horizon is subdivided into weeks, there are 52 planning periods in a year. The

CE implementation solves all our instances in matters of seconds and finds solutions close to or very close to optimal. Due to an imposed time limit of an hour, occasionally CE even finds better solutions than Cplex. Since CE is designed for single-objective unconstrained optimization problems, we included repair functions in our approach to guarantee feasible solutions. Given its solution speed and quality, CE is the preferred solution method for quickly analyzing multiple input scenarios. Also CE is well-suited for assessing the effect of changes in parameter values, for example holiday requests or illnesses.

This paper is structured as follows. Section 2 discusses literature on combinatorial optimization problems related to the SUAH problem, and on CE applications to combinatorial optimization problems. Section 3 gives a mathematical description of SUAH. Next, Section 4 describes the CE optimization method, how we employ it to solve the SUAH problem, and further motivates why we use CE optimization by comparing it to other optimization methods. After this, Section 5 presents our CE implementation and Section 6 gives numerical results. Conclusions and recommendations are presented in Section 7.

2 Literature

This section discusses literature. To our knowledge, there is no literature that considers both cost-efficient staffing and annualized hours as in the problem that is studied in this paper. However, separately both problems have been studied, as discussed in Section 2.1. After that, Section 2.2 discusses applications of Cross-Entropy optimization to combinatorial optimization problems.

2.1 Staffing under annualized hours literature

This section first discusses literature on staffing problems. After that, annualized hours literature is discussed.

The classical staffing problem as in [33] determines the number of employees needed to cover a given workload, where employees are considered equal, whereas we consider employees with different contract-sizes. In [6, 11, 39] staffing problems in production planning settings are studied, which are modeled as two-stage processes. The first stage decides on the necessary capacity investment. After that, when demand realizations become known, the second stage decides on capacity allocation. These papers focus mainly on profits that are induced by production capacities and demand. In [29] a staffing problem is studied in which employees can be hired and fired per period. A fixed demand is given per period and the number of hours an employee works is fixed per period. In addition, shortages are allowed, but lead to a penalty. The decisions in [29] are mainly about when

to hire and fire employees, without considering annualized hours. In [20] a staffing problem is solved where demands are expressed in shifts per period, under constraints on both the number and sequences of shifts employees can work, but without considering annualized hours.

Variants of the annualized hours problem are studied in [28, 27], where schedules are created explicitly, which differs from our problem. Also employees are considered equivalent in terms of number of hours and sequences of shifts they are allowed to work. In [34] shorter solving times are obtained after reformulating an MILP that models a problem closely related to ours. In [13, 15, 16] MILP approaches are proposed to solve the annualized hours problem for a fixed number of employees. Also the lower and upper bounds on the working hours per period are the same for all employees. In [2, 3] MILP approaches are proposed to solve an annualized hours problem, with demands expressed in the number of shifts, and constraints on the number and sequences of shifts employees can work. Again the number of employees is minimized, with the employees considered equivalent. A classification scheme for annualized hours problems is proposed by [14].

This paper considers cost-efficient staffing combined with annualized hours.

2.2 CE applied to combinatorial optimization problems

CE is applied to a variety of combinatorial optimization problems. Some of these problems are closely related to our problem. In [19] CE is used to solve a single-dimensional knapsack problem, with ρ dependent on t, and a 'repair' strategy is applied to 'repair' infeasible sample solutions. In [9] CE is again applied to the single-dimensional knapsack problem, but then with setups: a fixed cost is incurred if an item is selected and a variable profit is earned depending on the 'integer' usage of this item. Here, a repair strategy is also used to deal with sample solutions that do not satisfy all the problem constraints. As indicated in Section 4, in [7] large-scale capacitated facility location problems are solved by CE. This problem is similar to ours, see Section 3. Although literature does not report on CE applications to solve the *multidimensional* knapsack problem, the MATLAB File Exchange Central holds a script for this problem [32], see Section 6.

Applications of CE to combinatorial optimization problems that are less related to our problem are found in, e.g., [8] that solves a capacitated lot-sizing problem with setup times. Moreover, [35, 18, 38, 25] solve traveling salesman problems (TSP) using CE, and [36, 18, 37] solve the max-cut problem using CE. In [36], a variant of CE is applied, with the updating rule (18), and applications to numerous combinatorial optimization problems are also found in this paper. These include, single-dimensional knapsack, multiple knapsack, bipartite matching, satisfiability, and counting the number of feasible solutions for integer programming. In [1, 10, 26] network re-

liability optimization problems are solved by using CE. A multi-objective optimization problem is solved in [5].

3 Problem description

Consider a workforce planning problem over a finite planning horizon T. Employing staff is subject to cost and availability constraints. Employees, indexed by $i = 1, \ldots, m$, are employed throughout the whole planning horizon at a cost c_i. The number of working hours employee i is available during T is w_i. Furthermore, the time horizon T is discretized into time periods, indexed by $j = 1, \ldots, n$, in which there is a demand for a number of working hours, $d_1, \ldots, d_n$. In addition, we have lower and upper bounds on the number of hours employee i *should* and *can* work during period j, denoted by l_{ij} and u_{ij}, respectively. Here, l_{ij} represents the minimal number of hours employee i is paid for and u_{ij} represents the maximum number of hours employee i is available in period j. It is assumed that $u_{ij} \geq l_{ij} \geq 0$. The objective is to select a subset of the employees and determine their working hours during every period j, such that demand is met in each period, and the number of hours the selected employees work is within the capacity restrictions. Furthermore, the cost of the employees should be minimized. We refer to this problem as the staffing under annualized hours (SUAH) problem.

Let x_i indicate whether employee i is part of the workforce ($x_i = 1$) or not ($x_i = 0$), and let w_{ij} denote the number of hours employee i works in period j. The objective is to minimize the cost of the employed workforce:

$$\sum_{i=1}^{m} c_i x_i. \tag{1a}$$

Three constraints are implied on the selection of employees. First, in every period j demand has to be met:

$$\sum_{i=1}^{m} w_{ij} \geq d_j \quad j = 1, \ldots, n. \tag{1b}$$

Second, the number of hours an employee works throughout the planning horizon equals w_i if the employee is part of the workforce, and 0 otherwise:

$$\sum_{j=1}^{n} w_{ij} = w_i x_i \quad i = 1, \ldots, m. \tag{1c}$$

Third, per period, per employee, w_{ij} must lie within the given lower and upper bounds, if the employee is part of the workforce, and equal 0 otherwise:

$$l_{ij} x_i \leq w_{ij} \leq u_{ij} x_i \quad i = 1, \ldots, m;\ j = 1, \ldots, n. \tag{1d}$$

Finally, for x_i we have:

$$x_i \in \{0,1\} \quad i = 1, \ldots, m. \tag{1e}$$

Model (1) now states an MILP formulation of SUAH.

Note that the multidimensional knapsack problem is the special case of (1) where $l_{ij} = u_{ij}$. Also note that this problem is closely related to the capacitated facility location problem (CFLP), if we regard the facilities and destinations of CFLP as the employees and planning periods of SUAH. However, there are two important differences. First, in SUAH there is no transportation cost c_{ij}. Second, and more importantly, SUAH has lower (l_{ij}) and upper (u_{ij}) bounds on the amount transported from location i to destination j. Adding these bounds to CFLP increases the complexity. For CFLP to be feasible it is sufficient to have $\sum_{i=1}^{m} w_i x_i \geq \sum_{i=j}^{n} d_j$, whereas for SUAH it is not.

4 Solution Method

This section discusses the Cross-Entropy optimization (CE) method. In addition to solving and modeling the staffing under annualized hours (SUAH) problem as MILP as in formulation (1), we apply CE to solve SUAH. Section 4.1 gives a general description of the CE optimization technique, and, after that, Section 4.2 motivates why we use CE to solve SUAH. The details of our CE implementation are discussed in Section 5.

4.1 Cross-Entropy optimization

The CE method was initially developed during the late 1990s [35]. Loosely speaking, CE is a self-learning importance sampling (IS) method. Importance sampling is a well-known technique for rare event simulation. However, CE is also applied to many combinatorial optimization problems.

A general introduction to CE, that is more elaborate than presented here, is found in [18, 38]. CE is an iterative method that generates samples of solutions, and in every iteration based on 'good' solutions, the sampling distribution's parameters are updated such that better solutions are generated in the next iteration.

To make this more formal, we closely follow the approach in [18]. In this paper we consider the general 0-1 binary minimization problem, with performance evaluation function $S : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \{0,1\}^m$ represents the feasible region. We want to minimize S over $\mathcal{X}$, i.e., find x^* such that:

$$S(x^*) = \min_{x \in \mathcal{X}} S(x) = \gamma^*. \tag{2}$$

CE aims to find x^* by iteratively generating random samples from the family of random variables $X^{(v)}$ with probability distribution function $f(x; v), x \in$

$\mathcal{X}, v \in \mathcal{V}$, where $\mathcal{V}$ is a set of parameter values. In this paper we let $f(x; v)$ be the Bernoulli distribution:

$$f(x; v) = \prod_{i=1}^{m} v_i^{x_i}(1 - v_i)^{1-x_i}. \tag{3}$$

Now, for $x \in \mathcal{X}, v \in \mathcal{V}$, and $\gamma \in \mathbb{R}$, let:

$$l_v(\gamma) = \mathbb{P}_v\{S(X^{(v)}) \leq \gamma\} = \mathbb{E}_v I_{\{S(X^{(v)}) \leq \gamma\}} = \sum_{x \in \mathcal{X}} I_{\{S(x) \leq \gamma\}} f(x; v), \tag{4}$$

where the indicator function $I_A = 1$ if A occurs, and 0 otherwise.

CE iterates over the set $\mathcal{V}$ to estimate γ^* and x^*, and the corresponding 0-1 vector $v^* \in \mathcal{V}$, such that $X^{(v^*)} = x^*$. Hence, $X^{(v^*)}$ is a deterministic random variable with all mass at x^*. The estimation problem (4) is often referred to as the *associated stochastic problem* (ASP).

For N i.i.d. random variables $X_1, \ldots, X_N$ distributed as $X^{(v)}$, an unbiased estimator of $l_v(\gamma)$ is:

$$\frac{1}{N} \sum_{i=1}^{N} I_{\{S(X_i) \leq \gamma\}}. \tag{5}$$

However, when $\{S(X_i) \leq \gamma\}$ is a rare event, a huge number of samples has to be generated. An alternative to this, which is used in CE, is based on *importance sampling* (IS). Take a random sample $X_1, \ldots, X_N$ with a different density $g(x)$ on $\mathcal{X}$, and evaluate $l_v(\gamma)$ using the likelihood-ratio (LR) estimator:

$$\widehat{l_v(\gamma)} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(X_i) \leq \gamma\}} \frac{f(X_i; v)}{g(X_i)}. \tag{6}$$

The best way to estimate $l_v(\gamma)$, see [18], is to use a change of measure with density. The optimal $g^*(x)$ would be:

$$g^*(x) = \frac{I_{\{S(x) \leq \gamma\}} f(x; v)}{l_v(\gamma)}, \tag{7}$$

since inserting (7) in (6) gives:

$$\widehat{l_v(\gamma)} = l_v(\gamma). \tag{8}$$

This is infeasible as $g^*(x)$ depends on the unknown $l_v(\gamma)$. Furthermore, it is convenient to choose $g^*(x)$ from the family of probability distributions $f(x; v)$. The idea in CE optimization is to choose v such that the distance between $g^*(x)$ and $f(x; v)$ is minimized. A convenient measure of distance between two densities $g(x)$ and $h(x)$ is the CE distance, also referred to as

the *Kullback-Leibler* distance. The CE distance between $g(x)$ and $h(x)$ is defined as:

$$\begin{aligned}\mathcal{D}\left(g(x);h(x)\right) &= \mathbb{E}_{g(x)}\left[\log\left(\frac{g(x)}{h(x)}\right)\right] \\ &= \int g(x)\log g(x)dx - \int g(x)\log h(x)dx. \end{aligned} \tag{9}$$

Minimizing the CE distance between $g^*(x)$ and $f(x;v)$ over v is equivalent to:

$$\max_v \int_{x\in\mathcal{X}} g^*(x)\log f(x;v)dx. \tag{10}$$

Substituting $g^*(x)$ from (7) in (10) gives:

$$\max_v \int_{x\in\mathcal{X}} \frac{I_{\{S(x)\leq\gamma\}}f(x;u)}{l(\gamma)}\log f(x;v)dx, \tag{11}$$

which is equivalent to:

$$\max_v \mathbb{E}_u I_{\{S(\cdot)\leq\gamma\}}\log f(\cdot;v). \tag{12}$$

Given a sample $X_1,\ldots,X_N$, which are generated under pdf $f(\cdot;u)$, we can estimate v from:

$$\hat{v} = \arg\max_v \frac{1}{N}\sum_{i=1}^{N} I_{\{S(X_i)\leq\hat{\gamma}\}}\log f(X_i;v), \tag{13}$$

where we let $\hat{\gamma}$ be the $(1-\rho)$-quantile of the performances, i.e., sort the $S(X_i)$ on decreasing value: $S_{(1)},\ldots,S_{(N)}$ and let:

$$\hat{\gamma} = S_{(\lfloor(1-\rho)N\rfloor)}. \tag{14}$$

In [18] is shown that, if V is the set of Bernoulli(v) distributions, the CE distance (9) is minimized for:

$$\hat{v}_i = \frac{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}\}}X_{k,i}}{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}\}}}, \tag{15}$$

where $X_{k,i}$ and $\hat{v}_i$ denote the i-th element of X_k and $\hat{v}$, respectively.

Note that updating rule (15) appeals to intuition, since it sets $\hat{v}_i$ as the fraction of the number of times that element i is present in the $1-\rho$ sample quantile. Note that - except for the indicator functions - (15) equals the maximum likelihood estimator for $\hat{v}$.

In order to approximate γ^* and v^* we update γ and v iteratively as in equation (14) and (15), respectively.

Using these, we get the algorithmic representation of CE optimization as in Algorithm 4.1.

Algorithm 4.1. *Cross-Entropy algorithm (for optimization).*

1. Let $\hat{v}_0 = u$, for some initial distribution parameter u. Let N denote the sample size, and let $d \in \mathbb{N} \setminus \{0\}$, and $0 < \rho < 1$ be given. Set $t = 1$ (iteration counter).

2. Generate a sample $X_1, \ldots, X_N$ from the density $f(\cdot; \hat{v}_{t-1})$ and compute the sample $(1-\rho)$-quantile $\hat{\gamma}_t$ of the performances, as in (14)

3. Use the same sample $X_1, \ldots, X_N$ and choose $\hat{v}_t \in V$ such that the CE distance $D\left(f(\cdot; \hat{v}_{t-1}), f(\cdot; \hat{v}_t)\right)$ as defined in (9) between $f(\cdot; \hat{v}_{t-1})$ and $f(\cdot; \hat{v}_t)$ is minimized.

4. If:
$$\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-d}, \tag{16}$$
then stop. Otherwise set $t = t + 1$ and return to 2.

According to [8], CE generates a population of binary vectors under a Bernoulli distribution of parameter v, evaluates each binary solution in terms of objective function value, and updates v with the aim of generating a better population in the next iteration.

Often updating rule (15) is replaced by

$$\hat{v}_{t,i} = \lambda \frac{\sum_{k=1}^{N} I_{\{S(X_k) \leq \hat{\gamma}_t\}} X_{k,i}}{\sum_{k=1}^{N} I_{\{S(X_k) \leq \hat{\gamma}_t\}}} + (1-\lambda)\hat{v}_{t-1,i}, \tag{17}$$

for some $0 < \lambda < 1$, in order to prevent the CE method from converging too quickly [18]. In addition, [17, 31] let λ depend on t, i.e., replace λ by λ_t in (17), to achieve this. We want to prohibit that CE converges too quickly, because once an entry of $\hat{v}_t$ is fixed to 0 or 1, only sample solutions with (if the entry is fixed to 1) or without (if the entry is fixed to 0) the corresponding element are generated. Another way to control the convergence of CE is to let ρ depend on t [19]. In [17, 31] conditions for λ_t are presented for the CE method to convergence to an optimal solution of the combinatorial optimization problem asymptotically with probability 1. Furthermore, [17] presents some necessary and sufficient conditions on λ_t for $f(x; \hat{v}_t)$ to converge with probability 1 to a unit mass located at some solution candidate x.

In the literature several other improvements to CE exist. In [36] it is suggested to use

$$\hat{v}_{t,i} = \frac{\sum_{k=1}^{N} S(X_k) I_{\{S(X_k^t) \leq \hat{\gamma}_t\}} X_{k,i}}{\sum_{k=1}^{N} S(X_k) I_{\{S(X_k^t) \leq \hat{\gamma}_t\}}}, \tag{18}$$

instead of (15). It is argued in [36] that this updating rule is, in general, at least as fast and accurate as (15) and it has a stronger mathematical

foundation. Another improvement is described in [7]. There, in every iteration local search is applied to the best elements of the sample solution. According to [7], better quality solutions are obtained for the problem they study when embedding local search in CE, but at the cost of some additional computation time.

In its basic form CE cannot handle constraint optimization problems, and is used to solve single objective optimization problems. When applying CE to constrained combinatorial optimization problems, such as the knapsack problem or the staffing under annualized hours problem, the generation of infeasible intermediate or final solutions has to be prevented. There are two main approaches to deal with this. A first approach is to penalize infeasibility, see, e.g., [1, 10, 17]. A second approach is to repair infeasible solutions, see, e.g., [9, 19]. In our implementation, we choose to penalize infeasibility, which is discussed in detail in Section 5, next to a possible application of repair functions.

4.2 Solution strategy motivation

This section motivates why we use CE to solve the SUAH problem.

CE is a metaheuristic approach such as simulated annealing, genetic algorithms, and tabu search. These metaheuristics all iteratively try to find better solutions based on the current and previously found solutions. As discussed in Section 4.1, CE iteratively generates samples of solutions and updates sampling parameters based on the 'good' solutions such that in the next iteration better sample solutions are generated. Iterations are continued until the best solution does not change for a set number of iterations, after which CE terminates and returns the best solution found. Whereas most metaheuristics often jump from one solution to another, CE jumps between sampling parameter values.

Metaheuristics provide the best near optimal solutions for the multidimensional knapsack problem (MKP), see [22]. SUAH is a generalization of the widely studied MKP, since SUAH reduces to MKP by letting $l_{ij} = u_{ij}$ and rewriting model (1) by removing redundant constraints and replacing w_{ij} by l_{ij}. According to [22], the success of exact approaches, like ILP and constraint programming, to solve MKP is limited. Metaheuristic approaches, like a tabu search approach combined with dynamic programming provide the best near-optimal solutions for the larger MKP test instances (up to 30 constraints and a few hundred variables). Therefore we explore metaheuristics to solve SUAH.

CE has proven to be successful on problems closely related to SUAH. Although CE is not applied to MKP, it shows some success on variants of (single) knapsack problems, see Section 2.2. In addition, CE is successfully applied to the capacitated facility location problem [7], which is also closely related to SUAH. Hence we choose to apply CE to SUAH.

In our numerical experiments we use MILP that given sufficient computation time gives an optimal solution, which also enables us to report on the solution quality of CE.

5 Implementation

The objective of model (1) is to select the most cost-efficient set of employees that is able to cover demand. We use Cross-Entropy (CE) optimization to solve this problem. For our problem we use CE to select the items, which is the "hard" part of our problem. When the set of selected items is known, an easy network flow problem remains, see Section 5.1. Similar approaches are found in, e.g., [9, 8], where the idea is to reduce complexity by letting CE intelligently guess which 'items' to select. In Section 5.2 we describe how CE is used to select a set of employees, i.e., determine values for x_i. Note that, as outlined in Section 4, CE in its basic form is designed to solve single-objective unconstrained problems. In Section 5.3 we discuss how we use feasibility conditions to incorporate demand constraints. However, since this approach does not necessarily lead to feasible solutions either, which is needed for practical applications, Section 5.4 discusses how repair functions can be used to guarantee feasible solutions. Section 5.5 discusses some details of our implementation of CE in MATLAB.

5.1 Annualized hours for given employees

If the values of x_i are known, the annualized hours problem is easily solved, since for given values of x_i (1) can be rewritten such that there are no lower bounds on w_{ij}, except for nonnegativity. Let $M = \{i|x_i = 1\}$ and $\bar{w}_{ij} = w_{ij} - l_{ij}$, $\bar{u}_{ij} = u_{ij} - l_{ij}$, $\bar{w}_i = w_i - \sum_{j=1}^{n} l_{ij}$ and $\bar{d}_j = \bar{d}_j - \sum_{j=1}^{n} l_{ij}$ then (1) reduces to:

$$\text{find} \quad \bar{w}_{ij} \tag{19a}$$

$$\text{s.t.} \quad \sum_{i \in M} \bar{w}_{ij} \geq \bar{d}_j \quad j = 1, \ldots, n \tag{19b}$$

$$\sum_{j=1}^{n} \bar{w}_{ij} = \bar{w}_i \quad i \in M \tag{19c}$$

$$0 \leq \bar{w}_{ij} \leq \bar{u}_{ij} \quad i \in M;\ j = 1, \ldots, n. \tag{19d}$$

Instead of an MILP we now have an LP, which is solvable by polynomial time algorithms. Moreover, (19) is a network *feasibility* problem [21, 23], for which special purpose algorithms exist.

5.2 Initialization

This section describes how our CE method is initialized. In many CE implementations that involve binary variables, v_0 is set to $\left(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right)$, see, e.g., [8, 26]. However, in our implementation we choose to set

$$v_0 = \left(\frac{\sum_{j=1}^{n} d_j}{\sum_{i=1}^{m} w_i}, \ldots, \frac{\sum_{j=1}^{n} d_j}{\sum_{i=1}^{m} w_i} \right). \tag{20}$$

The fraction $\sum_{j=1}^{n} d_j / \sum_{i=1}^{m} w_i$ equals the *expected* number of items needed for a solution to be feasible, if we ignore per time period demand constraints and only look at total capacity and total demand. Therefore, every generated solution has *in expectation* the number of items needed for a solution to be feasible. This way, the CE implementation converges faster and the probability to get stuck at infeasible solutions decreases.

5.3 Feasibility conditions

First, after a set of items is selected, feasibility can be checked by solving the network feasibility problem (19). Although this problem is polynomially solvable, it has to be solved for at least the best ρ percent samples in every iteration of the CE algorithm, which is quite time consuming if it has to be done often.

Therefore, we incorporate two feasibility conditions on the annualized hours problem into the staffing problem. Firstly, the total permitted number of employee working hours should be larger than or equal to the total demand, which is expressed by:

$$\sum_{i=1}^{m} w_i x_i \geq \sum_{j=1}^{n} d_j. \tag{21}$$

Secondly, the maximum permitted number of employee working hours in every period j should be larger than or equal to d_j, which is expressed by:

$$\sum_{i=1}^{m} u_{ij} x_i \geq d_j \quad \text{for } j = 1, \ldots, n. \tag{22}$$

Unfortunately, conditions (21) and (22) are necessary, but not sufficient feasibility conditions. The only sufficient condition for a network problem to be feasible, which we know of, is to solve it using a network flow formulation or an LP formulation.

We incorporate (21) and (22) into the CE implementation, by penalizing violations in the objective function. The objective function is then given by:

$$S(x) = \sum_{i=1}^{m} c_i x_i + \beta_1 \left(\sum_{j=1}^{n} d_j - \sum_{i=1}^{m} w_i x_i \right)^+ + \beta_2 \sum_{j=1}^{n} \left(d_j - \sum_{i=1}^{m} u_{ij} x_i \right)^+, \tag{23}$$

where $(x)^+ = \max\{x, 0\}$ and $\beta_1, \beta_2 > 0$. The first part of the objective function represents the cost of the workforce, the second and the third part represent penalties caused by violating (21) and (22), respectively.

5.4 Repair functions

The approach of Section 5.3 does not guarantee solutions to be feasible, also large values of β_1, β_2 in (23) do not guarantee solutions to be feasible. The feasibility conditions (21) and (22) are necessary, but not sufficient. However, we can guarantee solutions to be feasible by adding a repair function that 'repairs' infeasible solutions. In particular, using the network flow feasibility problem (19) we can check whether a given solution is feasible. If it is not feasible we invoke a repair strategy.

For this, we introduce an extension to model (19) that incorporate slack variables δ_j in the demand constraints as follows:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} \delta_j && \text{(24a)}\\
\text{s.t.} \quad & \sum_{i \in M} \bar{w}_{ij} + \delta_j \geq \bar{d}_j \quad j = 1, \ldots, n && \text{(24b)}\\
& \sum_{j=1}^{n} \bar{w}_{ij} = \bar{w}_i \quad i \in M && \text{(24c)}\\
& 0 \leq \bar{w}_{ij} \leq \bar{u}_{ij} \quad i \in M;\ j = 1, \ldots, n && \text{(24d)}\\
& 0 \leq \delta_j \quad j = 1, \ldots, n. && \text{(24e)}
\end{aligned}$$

The objective of (24) represents the demand that the employees in M together are unable to cover.

Furthermore, let:

$$j' = \arg \max_{j=1,\ldots,n} \delta_j, \tag{25}$$

j' denotes the index of the demand constraint for which we have the largest slack. Let i':

$$i' = \arg \max_{i \notin M} \left\{ \frac{c_i}{u_{ij'} - l_{ij'}} \right\}. \tag{26}$$

From the set of employees not in M, employee i' can contribute to covering the demand in demand constraint j' most cost-efficiently.

Algorithm 5.1 represents the repair strategy we invoke.

Algorithm 5.1. *Repair strategy*

1. Let M be the solution of CE. If it is feasible: done. Otherwise, go to 2.

2. Let δ be the solution to (24). Let j' be the solution of (25), and i' be the solution of (26). Let $M' = M \cup \{i'\}$. If M' offers a feasible solution: done. Otherwise, re-iterate using M'.

5.5 Software implementation

The CE method is implemented in MATLAB. For a fair comparison between CE and (not fine-tuned) CPLEX, a MATLAB script from [32], designed to solve the basic multidimensional knapsack problem with CE, is adapted to make it suitable to solve SUAH. Our adaptations to [32] include incorporating initialization rule (20), and modifying the objective function into (23). In our implementation we choose to set:

$$\beta_1 = \beta_2 = \frac{2 + 2\sum_{i=1}^{m} c_i}{\max\limits_{i=1,\ldots,m} \max\limits_{j=1,\ldots,n} (l_{ij} + u_{ij})}, \tag{27}$$

which is analoguous to the (single) β parameter used in [32], so as to stay close to the basic CE implementation of [32]. The intuition behind these parameter values is that if $\max_{i=1,\ldots,m} \max_{j=1,\ldots,n} (l_{ij} + u_{ij})$ is small, it is more likely that solutions are obtained where $\sum_{i=1}^{m} u_{ij} x_i$ is smaller than d_j for some j. To prohibit CE from getting stuck at these solutions we want β_1, β_2 to be larger for instances where $\max_{i=1,\ldots,m} \max_{j=1,\ldots,n} (l_{ij} + u_{ij})$ is small. The other way around, when $\max_{i=1,\ldots,m} \max_{j=1,\ldots,n} (l_{ij} + u_{ij})$ is larger, β_1, β_2 are smaller, since for these instances it is less likely that solutions are obtained where $\sum_{i=1}^{m} u_{ij} x_i$ is smaller than d_j. Determining good values for β_1, β_2 is a trade-off. On the one hand, large β_1, β_2 make it more likely to end up with feasible solutions, but, on the other hand, they make it less likely to find a (near) optimal solution; larger β_1, β_2 'push' CE harder away from the boundary of the feasible region, where the optimal solutions lie.

6 Experimental results

This section discusses the experimental results. The CE implementation described in Section 5 is compared with an MILP implementation of (1). The MILP instances are encoded as MPS-files and solved with a pre-compiled CPLEX 11.0 binary. We let CPLEX stop when the instances are solved to optimality, that is when the integrality gap is less than 0.01%, or if the solving time exceeds 3600 seconds. We did not perform extensive tuning on the parameters of CPLEX. The auto-tuner of CPLEX did not suggest parameter values that would be useful in general for solving the SUAH instances. Furthermore, we did no fine-tuning of CE parameters such that we compare two not fine-tuned implementations. All experiments are performed on an Intel Centrino Duo CPU 2.20 GHz, 3.0 GB of RAM.

Section 6.1 discusses the instances on which we test both implementations. After that, Section 6.2 and Section 6.3, evaluate the solving time and the solution quality of both implementations, respectively. As discussed in Section 5 we guarantee feasible solutions using the repair function discussed in Section 5.4 could be used.

6.1 Test instances

The implementation is tested on generated instances. We generate multidimensional knapsack problem instances that are transformed into SUAH instances. The objective function of a multidimensional knapsack *covering* problem is equal to (1a) and it has n constraints of the form:

$$a_{1j}x_1 + a_{2j}x_2 + \ldots + a_{mj}x_m \geq d_j \quad j = 1, \ldots, n. \tag{28}$$

The multidimensional knapsack instances are generated with the method of [12]. The values of a_{ij} are discrete uniform random numbers from $[0, 1000]$. Furthermore, $d_j = \alpha \sum_{i=1}^{m} a_{ij}$, where α is a *tightness* ratio specifying approximately the fraction of the total number of items that are needed for a solution to be feasible. We generate problem instances with the number of variables m set to $10, 20, \ldots, 100$, and the number of constraints n set to $5, 10, \ldots, 50$. For every m-n combination 30 problem instances are generated. Just as in [12], we have $\alpha = \frac{1}{4}$ for the first ten problems instances, $\alpha = \frac{1}{2}$ for the next ten instances, and $\alpha = \frac{3}{4}$ for the remaining ten instances. The objective function coefficients c_i are generated as:

$$c_i = \sum_{j=1}^{n} \frac{a_{ij}}{n} + 500q_i \quad i = 1, \ldots, m \tag{29}$$

where q_i is a real uniform random number from $(0, 1)$. According to [12] instances where the c_i and the a_{ij} are correlated are harder to solve than when they are not correlated. We generate ten SUAH instances from every multidimensional knapsack instance by setting $l_{ij} = (1 - p)a_{ij}$ and $u_{ij} = (1 + p)a_{ij}$ for $p = \frac{1}{10}, \frac{2}{10}, \ldots, 1$, and setting $w_i = \sum_{j=1}^{n} a_{ij}$. Here, p is the annualized hours parameter indicating the percentage we are allowed to over-staff or under-staff in a planning period. In total, we thus generate $10 \cdot 10 \cdot 30 \cdot 10 = 30000$ instances.

We generate these instances instead of the instances of [12] that are made publicly available [4], since the instances we generate better fit with our underlying practical application. In the SUAH problem it is typical to have up to 100 employees (variables) and up to 50 planning periods (constraints). For example, if a planning period is a week and the planning horizon is a year, we have 52 planning periods in the planning horizon.

6.2 Solving time

This section compares solving times of both implementations. We examine the effect of the number of variables m, the number of constraints n, the density parameter α, and the annualized hours parameter p on the solving time of both CE and MILP.

6.2.1 Effect of m on solving time

Figure 1 shows the effect of the number of *variables* on solving times of CE and CPLEX for $n = 40$. In the left figure solving times are averaged over α and p, in the figure on the right medians are computed.

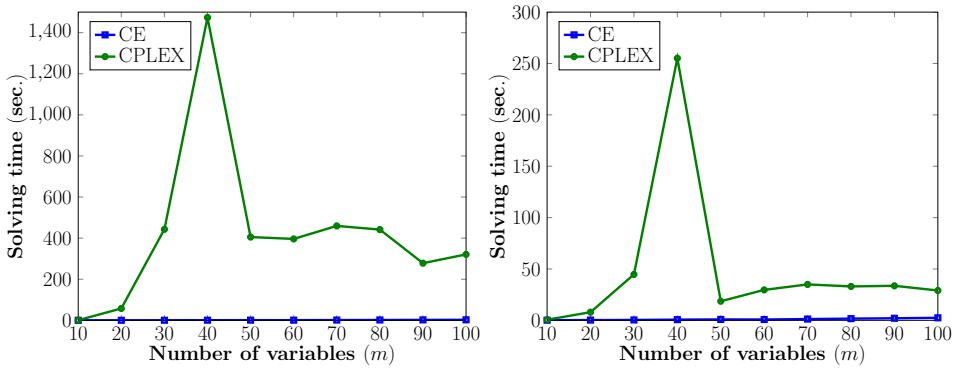

Figure 1: Mean (left) and median (right) solving time as function of m for $n = 40$

From Figure 1 we observe that CPLEX needs far more solving time than CE. Interesting to note is that for $m = 40$ CPLEX needs on average the most time. We think that for small m CPLEX is able to relatively quickly obtain a solution, since the number of solutions is limited, and that for larger m ($m \geq 50$) there are many good solutions, which makes it easier for CPLEX to find one. Also note that the average solving time of CPLEX is strongly influenced by a small number of instances that are for CPLEX hard to solve; the difference between CE and CPLEX is smaller when we look at the medians. We also observe that the CE solving time is almost constant. For other values of m, for which the graphs are not shown here, similar effects were observed.

6.2.2 Effect of n on solving time

Figure 2 shows the effect of the number of *constraints* on the solving time of CE and CPLEX for $m = 30$. Again, in the left figure solving times are averaged over α and p, and the right figure shows medians.

Figure 2 shows that both the mean and median solving time of CPLEX increases with the number of constraints, which we think is caused by the

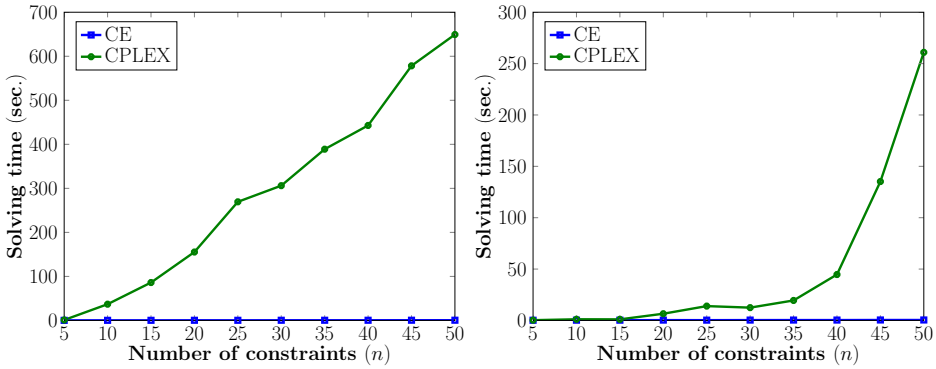

Figure 2: Mean (left) and median (right) solving time as function of n for $m = 30$

increasing computation time of the simplex method for an increasing number of constraints. For CE both the mean and the median solving time stay about the same when the number of constraints increases, which is logical since m only influences the computational effort needed to compute (23), which is, from a computational point, only a minor part of the CE implementation.

6.2.3 Effect of α on solving time

Table 1 shows the effect of the density parameter α on the solving time required by both CE and MILP.

Table 1: Effect of α on solving time

	solving time CE				solving time MILP			
α	min	max	mean	median	min	max	mean	median
1/4	0.1	3.7	0.9	0.7	0.0	3600.0	463.1	19.8
1/2	0.1	5.5	1.1	0.9	0.0	3600.0	293.1	7.1
3/4	0.1	4.1	0.9	0.7	0.0	3600.0	98.1	4.2

Table 1 shows that CE needs the most time to solve instances with $\alpha = \frac{1}{2}$. For these instances about half of the total items are selected in the optimal solution, and the number of ways to select x items out of a set of y items is the largest for $x = \frac{1}{2}y$. Hence, CE in expectation needs more iterations before no improvements are found anymore. The solving time CPLEX needs decreases for increasing α. For smaller α less items are selected in the optimal solution, which implies a smaller objective function value and hence the margin for the integrality gap is smaller. We think this makes it for CPLEX harder to find and return a solution for smaller α.

6.2.4 Effect of p on solving time

Figure 3 shows the effect of the value of the annualized hours parameter p on the solving time of CE and CPLEX. In the left figure solving times are averaged over m, n, and α, and in the right figure medians are computed.

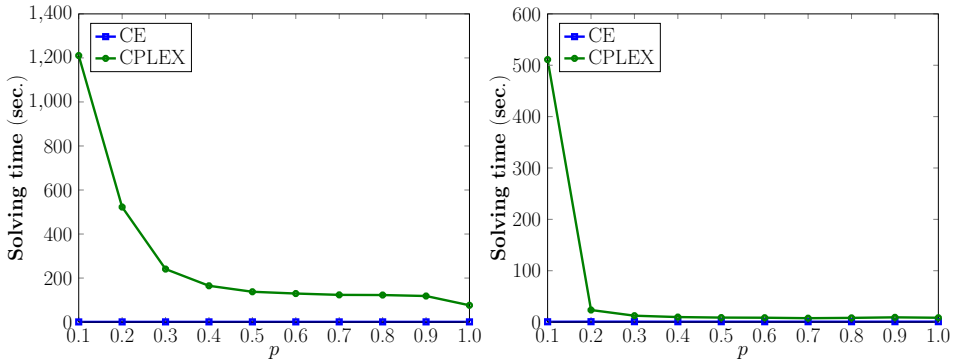

Figure 3: Mean (left) and median (right) solving time as function of p

From Figure 3 it is observed that the MILP instances are solved faster as p increases. We think this is a consequence of the larger difference between l_{ij} and u_{ij} in the instances where p is larger. In these cases, the bounds on w_{ij} are less tight, making it easier for CPLEX to find a solution. For CE both the mean and median solving time remain approximately the same for increasing values p. This is logical since the computational effort of CE is independent of the *values* of l_{ij} and u_{ij}.

6.2.5 Unsolved MILP instances

The number of MILP instances CPLEX is unable to solve within an hour is summarized in Table 2.

In Table 2 we see, for every m-n combination the number of unsolved instances. For small m ($m \leq 30$) and small n ($n \leq 10$) CPLEX solved most instances within the hour. However, for larger m and n many instances are not solved within the hour. Interesting to note is that for $m = 40$ CPLEX has the most unsolved instances. We think that for small m, CPLEX is able to quickly obtain a solution since the number of solutions is limited, and that for larger m ($m \geq 50$) there are many good solutions, which makes it easier for CPLEX to find a good solution, recall our discussion in Section 6.2.1.

6.3 Solution quality

This section compares the quality of the solutions produced by CE and CPLEX. The solution quality is defined as the objective value of the solution divided by the solution of the linear programming relaxation of the MILP.

Table 2: CPLEX unsolved instances

	m									
n	10	20	30	40	50	60	70	80	90	100
5	0	0	0	1	0	0	0	3	1	4
10	0	0	0	11	5	4	5	7	0	1
15	0	0	0	13	11	14	7	17	21	12
20	0	0	1	27	14	18	21	12	9	6
25	0	0	4	40	27	16	18	13	23	9
30	0	0	11	68	24	20	17	27	25	24
35	0	0	16	69	26	23	22	30	26	12
40	0	0	19	103	28	26	31	23	15	20
45	0	0	24	91	32	33	43	30	17	28
50	0	0	24	91	32	39	52	31	33	38

For less than 1% of the instances CE produces an infeasible solution. These instances are ignored during the comparison of the solution quality of CE and Cplex.

The largest observed integrality gap for Cplex is 9.6%. The average and median integrality gap are 0.06% and 0.01%, respectively. The worst performance of CE compared to the relaxation is 44.8%, which is relatively large, however on average it is 0.77% and the median is 0.12%. In practice, the error in the input data of the latter two figures is likely to be larger. Excluding the solutions for which the repair function needs to be applied, the worst performance of CE compared to the relaxation is 16.2%. More advanced repair functions probably improve on this worst performance figure. For some instances CE even performs better than Cplex. Since Cplex stops as soon as the integrality gap drops below 0.01%, CE sometimes finds a better solution. This solution is only slightly better, since the integrality gap is already smaller than 0.01%.

6.3.1 Effect of m and n on solution quality

In Figure 4, for both CE and Cplex, mean solution qualities are computed. The solution qualities are averaged over α and p. The left figure shows the effect of the number of variables m on the solution quality for $n = 15$, and the right figure shows, for $m = 30$, the effect of the number of constraints n on the mean solution quality.

From Figure 4 we observe that neither the number of variables nor the number of constraints has a clear effect on the solution quality. Although we observe that the solution quality of Cplex is on average better than the solution quality of CE, the effects are small.

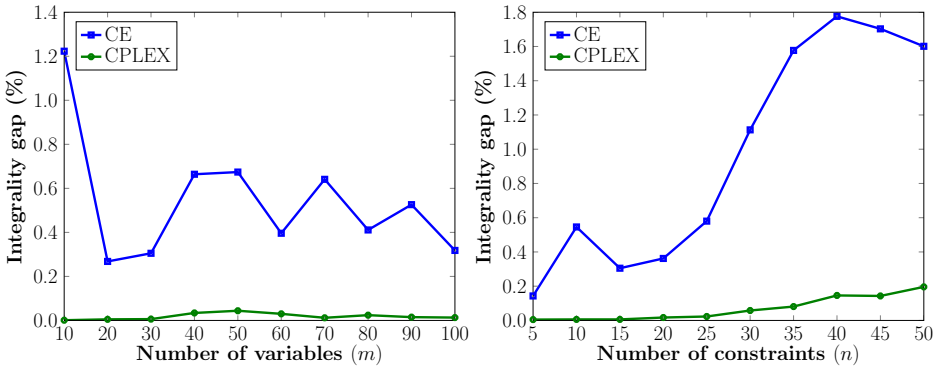

Figure 4: Mean solution quality as function of m for $n = 15$ (left) and as function of n for $m = 30$ (right)

6.3.2 Effect of α on solution quality

Table 3 shows the effect of the density paramter α on the solution quality. For each value of α the best, worst, mean, and median solution quality are shown.

Table 3: Effect of α on solution quality (integrality gap)

	CE				MILP			
α	best	worst	mean	median	best	worst	mean	median
0.25	0.0%	44.8%	1.3%	0.2%	0.0%	9.6%	0.13%	0.0%
0.50	0.0%	23.0%	0.6%	0.1%	0.0%	2.7%	0.03%	0.0%
0.75	0.0%	18.3%	0.4%	0.1%	0.0%	1.0%	0.01%	0.0%

As Table 3 shows the solution quality increases as α increases if looking at the worst and mean solution quality. We think this is caused by the fact that for larger α more items need to be selected in the optimal solution, which gives more freedom in the problem instance and makes it easier to find a good solution.

6.3.3 Effect of p on solution quality

In Figure 5 solution qualities are averaged over m, n, and α, and shown as a function of the annualized hours parameter p.

From Figure 5 we observe that the solution quality is better for increasing p for both CE and Cplex. We think this is a consequence of the larger difference between l_{ij} and u_{ij} in the instances where p is larger. In these cases, the bounds on w_{ij} are less tight. Provided that total capacity exceeds total demand, this makes it easier to find a solution, since there are less restrictions on the distribution of total capacity over all planning periods.

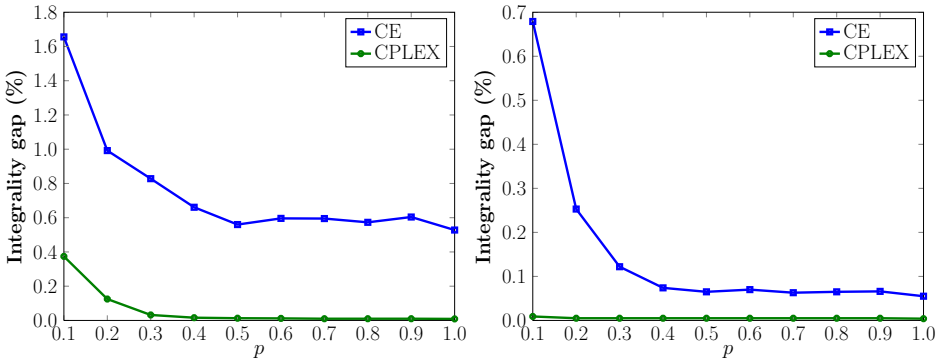

Figure 5: Mean (left) and median (right) solution quality as function of p

7 Conclusions and discussion

This paper studies the staffing under annualized hours (SUAH) problem. SUAH determines the optimal contract-mix of a workforce while applying annualized hours. We investigated the potential of Cross-Entropy (CE) optimization, a technique known from rare event simulation, to solve the SUAH problem. For this, we implemented CE in MATLAB. Our CE implementation includes a penalty function and a repair function to guarantee solutions to be feasible. In addition to the CE implementation, we modeled the SUAH problem as an MILP that is solved by CPLEX.

For practical applications, opportunities for improving our CE implementation exist. First, simulations to optimize the penalty parameters of the model could be run. In addition, improved or alternative penalty functions may be implemented. Second, updating schemes for the model parameters λ_t, ρ_t and N_t could be used. In [19, 17, 31] some success with this is mentioned, and conditions on λ_t for asymptotic optimality of CE are derived in [17, 31]. We implemented updating rule (18) of [36], but this had no noticeable positive effect for our CE implementation.

In total we generated 30000 test instances. We observe that CE solves the instances much faster than CPLEX. CE is able to produce high-quality solutions in matters of seconds. For most instances CPLEX produces the best solution, but for many instances it requires much more time to solve them. Moreover, for 1738 instances (5.8%), CPLEX is not able to find the optimal solution within our time limit of one hour. Especially for larger instances that are in size comparable to real-life instances, CPLEX requires much time to solve them, if solved at all within an hour. This makes our CE implementation well-suited for data validation and scenario-analysis, like evaluating the consequences of holiday requests and last-minute illnesses. In addition, for applications that have the SUAH problem, or a related problem, as a subproblem it is worthwhile to consider CE to solve these subproblems.

Acknowledgments

The authors would like to thank Paul Bakker, Johann Hurink, Werner Scheinhardt, and Bart Veltman for helpful discussions. In addition, the authors would like to thank the anonymous referees for their useful comments.

This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

References

[1] Fulya Altiparmak and Berna Dengiz. A cross entropy approach to design of reliable networks. *European Journal of Operational Research*, 199(2):542–552, 2009.

[2] Carlos S. Azmat, Tony Hrlimann, and Marino Widmer. Mixed integer programming to schedule a single-shift workforce under annualized hours. *Annals of Operations Research*, 128(1):199–215, 2004.

[3] Carlos S. Azmat and Marino Widmer. A case study of single shift planning and scheduling under annualized hours: A simple three-step approach. *European Journal of Operational Research*, 153(1):148–175, 2004.

[4] J. E. Beasley. Multidimensional knapsack problem instances [www page]. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`, Retrieved: April 2012.

[5] James Bekker and Chris Aldrich. The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, 211(1):112–121, 2011.

[6] Ebru K. Bish and Qiong Wang. Optimal investment strategies for flexible resources, considering pricing and correlated demands. *Operations Research*, 52(6):954–964, 2004.

[7] M. Caserta and E. Quionez Rico. A cross entropy-based metaheuristic algorithm for large-scale capacitated facility location problems. *Journal of the Operational Research Society*, 60(10):1439–1448, 2009.

[8] M. Caserta and E. Quionez Rico. A cross entropy-lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers & Operations Research*, 36(2):530–548, 2009.

[9] M. Caserta, E. Quionez Rico, and A. Mrquez Uribe. A cross entropy algorithm for the knapsack problem with setups. *Computers & Operations Research*, 35(1):241–252, 2008.

[10] Marco Caserta and Marta Nodar. A cross entropy based algorithm for reliability problems. *Journal of Heuristics*, 15(5):479–501, 2009.

[11] Jiri Chod and Nils Rudi. Resource flexibility with responsive pricing. *Operations Research*, 53(3):532–548, 2005.

[12] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[13] A. Corominas, A. Lusa, and R. Pastor. Using MILP to plan annualised working hours. *The Journal of the Operational Research Society*, 53(10):1101–1108, 2002.

[14] Albert Corominas. Characteristics and classification of the annualised working hours planning problems. *International Journal of Services Technology and Management*, 5:435–447, 2005.

[15] Albert Corominas, Amaia Lusa, and Rafael Pastor. Planning annualised hours with a finite set of weekly working hours and joint holidays. *Annals of Operations Research*, 128(1):217–233, 2004.

[16] Albert Corominas, Amaia Lusa, and Rafael Pastor. Using a MILP model to establish a framework for an annualised hours agreement. *European Journal of Operational Research*, 177(3):1495 – 1506, 2007.

[17] Andre Costa, Owen Dafydd Jones, and Dirk Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573 – 580, 2007.

[18] Pieter-Tjerk de Boer, Dirk Kroese, Shie Mannor, and Reuven Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[19] Libao Deng, Liyan Qiao, and Xiyuan Peng. A directed quantile cross-entropy method for 0/1 knapsack problems. In *Electronic Measurement Instruments, 2009. ICEMI '09. 9th International Conference on*, pages 4–319 – 4–322, 2009.

[20] Hamilton Emmons and Du-Shean Fuh. Sizing and scheduling a full-time and part-time workforce with off-day and off-weekend constraints. *Annals of Operations Research*, 70(0):473–492, 1997.

[21] Hassan Salehi Fathabadi and Mehdi Ghiyasvand. A new algorithm for solving the feasibility problem of a network flow. *Applied Mathematics and Computation*, 192(2):429–438, 2007.

[22] Arnaud Fréville. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.

[23] Mehdi Ghiyasvand. A new approach for computing a most positive cut using the minimum flow algorithms. *Applied Mathematics and Computation*, 176(1):27–36, 2006.

[24] Bernard Grabot and Agns Letouzey. Short-term manpower management in manufacturing systems: new requirements and DSS prototyping. *Computers in Industry*, 43(1):11–29, 2000.

[25] Bjarne E. Helvik and Otto Wittner. Using the cross-entropy method to guide/govern mobile agent's path finding in networks. In *Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications*, MATA '01, pages 255–268, London, UK, 2001. Springer-Verlag.

[26] Kin-Ping Hui, Nigel Bean, Miro Kraetzl, and Dirk P. Kroese. The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134(1):101–118, 2005.

[27] R. Hung. Scheduling a workforce under annualized hours. *International Journal of Production Research*, 37(11):2419–2427, 1999.

[28] Rudy Hung. A multiple-shift workforce scheduling model under annualized hours. *Naval Research Logistics (NRL)*, 46(6):726–736, 1999.

[29] Yongjian Li, Jian Chen, and Xiaoqiang Cai. An integrated staff-sizing approach considering feasibility of scheduling decision. *Annals of Operations Research*, 155(1):361–390, 2007.

[30] MacMeeking. Why Teso's new composite distribution needed annual hours. *International Journal of Retail and Distribution Mangement*, 23(9):36–38, September 1995.

[31] L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134(1):201–214, 2005.

[32] Sebastien Paris. Multi-knapsack solver [www page]. `http://www.mathworks.com/matlabcentral/fileexchange/20436-multi-knapsack-solver`, Retrieved: April 2012.

[33] Michale L. Pinedo. *Planning and Scheduling in Manufacturing and Services, 2nd Edition*. Springer, New York, NY, USA, 2009.

[34] Chase Rainwater, Joseph Geunes, and H. Edwin Romeijn. The generalized assignment problem with flexible jobs. *Discrete Applied Mathematics*, 157(1):49 – 67, 2009.

[35] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[36] Reuven Rubinstein. Semi-iterative minimum cross-entropy algorithms for rare-events, counting, combinatorial and integer programming. *Methodology and Computing in Applied Probability*, 10:121–178, 2008.

[37] Reuven Y. Rubinstein. Cross-entropy and rare events for maximal cut and partition problems. *ACM Transactions on Modeling and Computer Simulation*, 12(1):27–53, January 2002.

[38] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[39] Jan A. Van Mieghem and Maqbool Dada. Price versus production postponement: Capacity and competition. *Management Science*, 45(12):1639–1649, 1999.