Learning and Anticipation in Online Dynamic Optimization with Evolutionary Algorithms: The Stochastic Case

Peter A.N. Bosman
Centre for Mathematics and Computer Science
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
Peter.Bosman@cwi.nl

Han La Poutré
Centre for Mathematics and Computer Science
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
Han.La.Poutre@cwi.nl

ABSTRACT

The focus of this paper is on how to design evolutionary algorithms (EAs) for solving stochastic dynamic optimization problems online, i.e. as time goes by. For a proper design, the EA must not only be capable of tracking shifting optima, it must also take into account the future consequences of the evolved decisions or actions. A previous framework describes how to build such EAs in the case of non-stochastic problems. Most real-world problems however are stochastic. In this paper we show how this framework can be extended to properly tackle stochasticity. We point out how this naturally leads to evolving strategies rather than explicit decisions. We formalize our approach in a new framework. The new framework and the various sources of problem–difficulty at hand are illustrated with a running example. We also apply our framework to inventory management problems, an important real–world application area in logistics. Our results show, as a proof of principle, the feasibility and benefits of our novel approach.

Categories and Subject Descriptors

F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Online Computation*; G.1 [**Numerical Analysis**]: Optimization; I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance, Experimentation

Keywords

Evolutionary Algorithms, Dynamic Optimization, Online Optimization, Stochastic Optimization, Anticipation

1. INTRODUCTION

Optimization problems in real–world settings are often dynamic, i.e. they change with time. Typical examples are vehicle routing [14, 19], inventory management [16, 18] and scheduling [1, 8]. Since the dynamic changes are typically unknown beforehand the problem generally has to be solved online, i.e. as time goes by [4, 9, 12, 20].

The problems to be solved, even for a single point in time, are often hard. Also, there is typically not much time between two subsequent decision times. For these reasons, restarting optimization from scratch is often undesirable. The tracking of (near–) optima, once they have been found, is therefore desirable. To be able to do this, the optimization algorithm needs to have a proper degree of adaptivity. Evolutionary algorithms (EAs) [11] are good candidates as they employ a set of solutions rather than a single solution. Adaptivity is then a virtue of issues such as maintaining diversity around (sub)optima and continuously searching for new regions of interest that may appear over time [7].

Tracking optima alone is not enough however. To solve dynamic optimization problems online, a myopic, i.e. "nearsighted", approach is often taken. The quality of a decision is then taken only to be how good it is in the current situation. This approach however is blind to the important issue of time–dependence: decisions taken now have consequences in the future. Because of this, a myopic approach can perform poorly in the long run. To properly tackle time–dependence, anticipation of future situations is needed to be able to make well–informed decisions [5, 6, 8, 21]. In problems in practice, this is typically required. Consider the problem of vehicle routing. Traditionally, routes are planned to maximize profit on a daily basis. However, poor quality of service (e.g. not being on time) for a specific customer may decrease the number of orders from that customer over the next days. Hence, profit is not only determined by the efficiency of today's routing, but also by the impact the resulting quality of service has on future events.

Previous work investigated explicit modeling of anticipation for online dynamic optimization, but only for non–stochastic problems [5]. Real–world online dynamic optimization problems are however often stochastic. We will point out that replacing the distributions that cause the stochasticity with their expected values to obtain a non–stochastic version of the problem does not work in general, i.e. it is *not effective*. Other previous work tackles stochasticity in dynamic optimization problems properly, but requires optimization from scratch to be performed many times for each new decision [2, 3, 10], i.e. it is *not efficient*.

It is the focus of this paper to obtain the best of both worlds, i.e. to an approach that is both *effective* and *efficient*. We will specifically focus on EAs and show how

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

their adaptive and evolving nature can be matched to the ongoing changes in stochastic online dynamic optimization problems, thereby exploiting the potentials of EAs for dynamic optimization. The approach is richer than just using expected values, while at the same time not requiring optimization from scratch for every new decision. We point out how this approach naturally leads to evolving strategies rather than explicit decisions and present a framework for our proposed approach. The new framework and the various sources of problem–difficulty at hand are illustrated with a simple running example. We also apply our framework to inventory management problems, an important real–world application area in logistics. Our results show how our novel approach can indeed be both effective and efficient.

The remainder of this paper is organized as follows. We discuss the necessity of anticipation in online dynamic optimization in Section 2. In Section 3 we then discuss how optimization can be performed in an online setting while making use of anticipation. We describe our new framework for solving online stochastic dynamic optimization problems with EAs in Section 4 and apply it to inventory management problems in Section 5. We conclude the paper with a summary and some final remarks in Section 6.

2. ANTICIPATION

Let $\boldsymbol{x}$ denote the variables to be optimized. We will often refer to choosing a configuration for these variables as taking a decision in online dynamic optimization. Mathematically defined, dynamic optimization is to optimize a functional

$$\max_{\boldsymbol{x}(t)} \left\{ \int_{0}^{t^{\text{end}}} \mathfrak{F}_{\boldsymbol{\gamma}(t)} \left(\boldsymbol{x}(t) \right) dt \right\} \tag{1}$$

where $\mathfrak{F}$ is a function of $\boldsymbol{x}$ and has dynamically changing parameters $\boldsymbol{\gamma}$. Note that in the discrete case, the integral in Equation 1 is replaced by a discrete sum. Function $\mathfrak{F}$ can be seen as the *real world*. Solving this problem online means that at any point in time t^{now}, function $\mathfrak{F}$ cannot be evaluated for any $t > t^{\text{now}}$.

The myopic approach to this problem amounts to:

$$\max_{\boldsymbol{x}(t^{\text{now}})} \left\{ \mathfrak{F}_{\boldsymbol{\gamma}(t^{\text{now}})} \left(\boldsymbol{x}(t^{\text{now}}) \right) \right\} \tag{2}$$

An important problem is that decisions may have future consequences. This means that $\boldsymbol{\gamma}(t^{\text{now}})$ may depend on previous decisions $\boldsymbol{x}(t), t < t^{\text{now}}$. A schematic illustration is given in Figure 1. If only the current situation is taken into account, the decision that immediately leads to the highest reward is optimal. This decision may however lead to a future in which lower profits can be obtained. If a suboptimal solution for the current situation is however taken, which typically corresponds to actions like making investments, a future may be created in which much higher rewards can be obtained. Integrated over the entire time–span, this suboptimal decision for the current situation may thus very well be the better choice. Moreover, it can be shown that the difference between the optimum when using the myopic approach compared to using anticipation can be arbitrarily big [5].

To prevent bad decisions due to the use of a myopic approach the decision for the current situation needs to be regarded simultaneously with future decisions in (near) future situations. Mathematically, this amounts to solving:

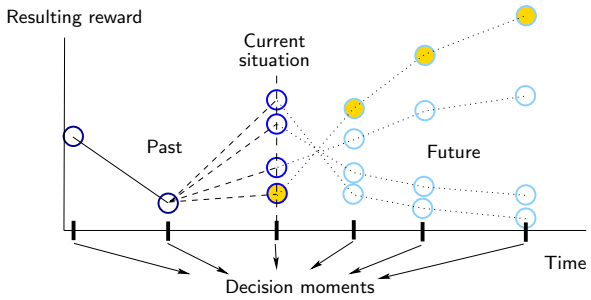

Figure 1: A schematic illustration of why anticipation may be necessary.

$$\max_{\boldsymbol{x}(t)} \left\{ \int_{t^{\text{now}}}^{t^{\text{end}}} \mathfrak{F}_{\boldsymbol{\gamma}(t)} \left(\boldsymbol{x}(t) \right) dt \right\} \tag{3}$$

Theoretically, this approach is clearly optimal because of the similarity with Equation 1. The problem is of course that this approach involves evaluating function $\mathfrak{F}$ for times beyond the current time, which is not possible in online optimization. The only way we can still take into account the future is to learn to predict it and use the predicted future instead. The nice thing about this approach is that under the condition of perfect prediction and the possibility of finding the optimum with the optimization algorithm of choice, optimal decisions can be taken. Summarizing, the approach is to build and maintain (i.e. learn online) an approximation of $\mathfrak{F}_{\boldsymbol{\gamma}(t)}$ and to optimize decisions for the present and for the approximated future simultaneously. In practice, it is typically not possible to optimize the future indefinitely, so only part of the approximated future will be optimized:

$$\max_{\boldsymbol{x}(t)} \left\{ \int_{t^{\text{now}}}^{\min\left\{t^{\text{now}}+t^{\text{plen}}, t^{\text{end}}\right\}} \hat{\mathfrak{F}}_{\boldsymbol{\alpha}} \left(t, \boldsymbol{x}(t) \right) dt \right\} \tag{4}$$

where $\boldsymbol{\alpha}$ are the parameters that pertain to the function class which we use to learn approximation $\hat{\mathfrak{F}}$. In logistics settings, typical examples of $\boldsymbol{\alpha}$ include the rate of customer demand or the parameters describing the distribution of customer demand. Function $\hat{\mathfrak{F}}$ can be seen as a *simulation* of the real world. Note that an apparent drawback is that if the consequences of decisions extend beyond the prediction length t^{plen}, optimal decisions can no longer be guaranteed.

3. OPTIMIZATION

From the previous section it has become clear that optimization of multiple decisions in a future time interval is required. The approach to how this can be done depends on whether the problem is stochastic.

3.1 Non–stochastic problems

In case of a non–stochastic problem, a future trajectory of the problem variables can be optimized directly. Because the problem changes in a fixed way, decisions can be planned ahead. Assuming, without loss of generality, a time–discretized representation with a discretized prediction interval of t^{pint}, optimization then regards a list of decisions $\boldsymbol{x}(t^{\text{now}}), \boldsymbol{x}(t^{\text{now}} + t^{\text{pint}}), \boldsymbol{x}(t^{\text{now}} + 2t^{\text{pint}}), \ldots, \boldsymbol{x}(t^{\text{now}} + N^{\text{future}} t^{\text{pint}})$ where N^{future} is the number of time–steps to optimize into the future, i.e. $N^{\text{future}} t^{\text{pint}} = t^{\text{plen}}$.

The adaptivity characteristic of EAs can be exploited for optimization in this case. Each genotype encodes the concatenation of all variables. As the problem changes with time, the decisions need to change accordingly. Under the assumption that the changes in the problem are not chaotic, adaptation by means of an EA will be more efficient than restarting optimization from scratch.

The use of a list of decisions was formalized in a framework and used in combination with EAs recently [5]. Initial results showed that under the assumption that the learning method that is used to compute $\hat{\mathfrak{F}}$ gives good approximations, this approach is able to tackle time–dependence and exhibit anticipation well.

3.2 Stochastic problems

If the problem is stochastic however, as most real–world problems are, a single list of decisions can no longer be used directly. To see why this is the case, consider the following very simple example pickup problem. A truck is located at $\text{TRUCK}(t)$ and a package appears at location $\text{PACKAGE}(t)$, both locations are 2D coordinates. It must now be decided whether to send the truck to go and pick up the package. If the package is not picked up, it disappears. Picking up the package pays a value of 1, but costs a value equal to the Euclidean distance that must be traveled. The number of packages is $n_{\text{packages}} = t^{\text{end}} + 1$, i.e. the time–steps are of size 1. A decision at time t is a binary variable $x(t) \in \{0, 1\}$ that indicates whether the package at time t will be picked up in the upcoming time unit ($x(t) = 1$). The stochasticity of this problem lies in the locations of the packages. Mathematically ($\sim$ means "distributed according to"):

$$\mathfrak{F}\left(x(t)\right) = \begin{cases} 1 - \parallel \text{PACKAGE}(t) - \text{TRUCK}(t) \parallel & \text{if } x(t) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\text{TRUCK}(t) = \begin{cases} \sim \mathcal{N}(0,1) \times \mathcal{N}(0,1) & \text{if } t = 0 \\ \text{PACKAGE}(t-1) & \text{if } t > 0 \text{ and } x(t-1) = 1 \\ \text{TRUCK}(t-1) & \text{otherwise} \end{cases}$$

Now consider a sequence of two decisions $x(t), x(t+1)$. Certainly, there is time–dependence in this problem: the decision to drive somewhere changes the location of the truck for the next time–step. Because of the stochasticity in the problem however, it doesn't make sense to plan $x(t+1)$ beforehand. This decision depends not only on the location of the truck in the next time–step, but also on the location of the package in the next time–step. For example, deciding to set $x(t+1) = 1$ is a bad choice if $\text{PACKAGE}(t+1)$ turns out to be far away from $\text{TRUCK}(t+1)$. Note that this is a very simple problem, meant to serve only as an illustration. There is no higher order of time–dependence such as a response of the system to drop packages in relation to decisions made earlier. Instead, packages are dropped off randomly.

3.2.1 Expected value

To still be able to optimize a single sequence of decisions if the optimization problem is stochastic, the stochasticity must be removed from the approximation. Note that the approximation can then no longer equal the actual problem.

A statistical approach to removing the stochasticity is to replace each random variable in the problem with its expected value. This approach has recently been applied, with good results, on a vehicle routing problem [6]. It is however important to realize the limitations of this approach. It only works well if the variance is small and the expected value is representative of the probability distribution. Regard the pickup problem with a factorized normal distribution with zero mean and unit variance for the package distribution:

$$\text{PACKAGE}(t) \sim \mathcal{N}(0,1) \times \mathcal{N}(0,1)$$

The expected value is $E[\text{PACKAGE}(t)] = (0, 0)$, i.e. the origin. If the vehicle at time t^{now} is located at the origin, a decision not to pick up the current package has an approximated profit of N^{future}. According to the approximation, each new package arrives at the origin and there will be no driving cost. A decision to pick up the current package on the other hand has a maximum approximated profit of $N^{\text{future}} + 1 - 2 \parallel \text{PACKAGE}(t^{\text{now}}) \parallel$. Thus, if the current package is within a distance of $\frac{1}{2}$ of the origin, a decision to pick up the package will be made. The truck will thus remain in the vicinity of the origin. Since the distribution generating the package is normally distributed, using the expected value will thus result in good decisions. Experimental verification is plotted in Figure 2. A simple EA is used with recombination as done in UMDA [17], i.e. compute for each bit the proportion of ones in the selected set of solutions and resample new solutions with this proportion. To prevent complete convergence, the proportions are enforced to remain in $[0.2, 0.8]$. The EA is allowed to run for 50 generations between two subsequent decision moments. Moreover, $N^{\text{future}} = 8$, implying that genotypes are bitstrings of length 8, and the results are averaged over 50 independent runs. For comparison, we also evaluated the use of a hillclimber. The hillclimber always performs pickup if the direct profit is larger than 0. This means that the vehicle may wander off far from the origin, making this a suboptimal strategy. The distribution of the packages is learned online using maximum–likelihood estimates. For the first few steps however there is not enough data to establish a reliable estimate. For this reason, the hillclimber strategy is used for the first 10 time steps. The results in Figure 2 (top lines in the graph) clearly show that the EA outperforms the hillclimber in the longer run.

Now consider the case where the package distribution consists of *four* normal distributions with unit variance, with one normal distribution located in each quadrant. Specifically, the means of these distributions are $(2, 2)$, $(-2, 2)$, $(-2, -2)$ and $(2, -2)$. Although the expected value is again at the origin, the density at the origin and its vicinity is quite low because it is beyond more than one standard deviation of each normal. Using the expected value in anticipation and a single decision–list leads to the same strategy as before. In this case however, this strategy is not a good one because only few packages will actually appear in the vicinity of the origin. Indeed, the results in Figure 2 show that the EA in this case performs much worse than the hillclimber.

3.2.2 Scenarios

Decision–trajectory optimization

As an alternative to removing the stochasticity from the problem, scenario–based optimization can be used. In this approach, multiple scenarios (i.e. events in the simulation) are sampled and a sequence of decisions is optimized for each scenario separately. To take a decision for the current situation, the decision that leads to the highest expected value of the profit (i.e. average) is then taken [10]. Other choices are

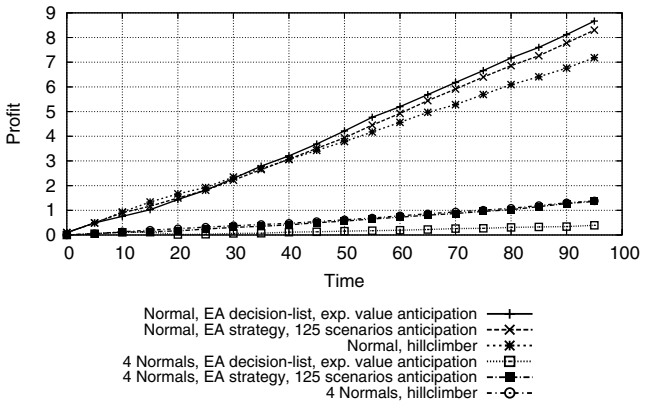

Figure 2: Profit obtained on the pickup problem with $N^{\text{future}} = 8$ and a population size of 120 for the EA; results are averaged over 50 runs.

possible as well, such as choosing the decision that leads to the smallest risk of having losses. It has been shown recently that this approach can provide high–quality solutions in the expected–value sense [15]. It is important though that the expected difference between the optimal choice for any scenario and the optimal expected–value choice doesn't become too large. It has already been argued that this assumption is satisfied in most real–world problems. It has also been proved to be the case for a real–world problem (packet scheduling) [15]. In addition, this approach is distribution–independent. It therefore doesn't have the drawback that the expected–value approach has of possibly misrepresenting the actual problem. Still, there are some important drawbacks to this approach.

First, let us mention that the application of this approach so far has been limited to sampling scenarios beforehand and then optimizing the future decisions in these scenarios [2, 3, 10]. In these papers, time–dependence is explicitly not considered in full, which is also stated by their authors. The type of time–dependence that is automatically tackled is the direct influence of one decision on another decision. For example, deciding to drive to customer 1 puts the truck at that location instead of the location where it would otherwise have stayed. What is not taken into account is the possible influence of a decision on the future response of the system, e.g. the stochastics. For example, deciding to drive to customer 1 may lead to a higher frequency of new orders from customer 1 in the near future. With the existence of time–dependence of this kind, sampling events in a scenario beforehand is unacceptable because the events may change as a result of decisions made.

The general approach requires a minor modification to allow for time–dependence in any of its forms to be tackled. Optimization with scenarios can be done by randomly choosing $N^{\text{scenarios}}$ random seeds and then optimizing the simulation for each random seed. This essentially makes the approximation (i.e. simulation) deterministic, allowing it to be re–evaluated under the exact same circumstances during optimization. The computational implications of this change are however not minor. Finding a solution (i.e. future trajectory) for each and every scenario requires solving an interactive problem (i.e. an online dynamic optimization problem without stochastics). To be able to do this efficiently, this requires complex algorithmic design. It is known that problems like this are hard. The field of online optimization itself is relatively young [20]. Alternatively, exhaustive search can be used, but this is very inefficient.

A second and very related drawback also shows that this approach is inherently more time–consuming. The use of the expected value can be seen as using only a single scenario. Hence, $N^{\text{scenarios}}$ as many evaluations are required when using scenarios. When re–optimization is performed, optimization has to be restarted $N^{\text{scenarios}}$ as many times.

A third drawback is that by optimizing the decisions directly, optimization from scratch at every decision moment is mandatory. For each new decision to be made for a new current situation, a new set of random seeds is chosen to obtain a new set of scenarios. These scenarios are therefore not related to the previously optimized scenarios. Also, a new decision trajectory needs to be found for each scenario. So not only is optimization per scenario complex as mentioned above, optimization is also required to be performed often and it needs to be done from scratch. This can be especially burdensome or even infeasible in an online time fashion because there is typically not weeks of time available between taking subsequent decisions and most of the underlying (combinatorial) optimization problems are at least $\mathcal{NP}$–hard. Because restarting optimization from scratch is required, the adaptivity of EAs cannot be exploited.

A fourth drawback is related to the third one. If the decision to be made concerns *continuous* (e.g. real–valued) variables, it is not possible to optimize decision trajectories for multiple scenarios and then choose the decision for the current situation with maximum average profit. For continuous decision variables it is not likely that optimal values will be the same under different scenarios. Discretization is then required, but doing this properly is hard.

Strategy optimization

Most above drawbacks can be overcome by using strategies instead of decision lists. By strategy we mean a function that, given the current time and situation, returns a decision. In the literature, such strategies are also sometimes called anticipative decision processes [15]. Take for example the pickup problem. A strategy could for instance be a rule that dictates that a pickup is only performed if the package is within a certain distance of the truck or within a certain distance of some other location. This distance is then a parameter of the strategy that can be optimized. The use of strategies has several important advantages.

A first advantage is that only one strategy needs to be optimized. The evaluation of the quality of a strategy encompasses its application to a set of scenarios, but because a strategy describes what to do in any given situation, it does not have to be re–optimized for each scenario. Hence, the adaptivity characteristic of EAs can now be exploited by optimizing the parameters of the strategy instead of the individual decisions directly. Certainly, when the problem changes, the strategy may need to change as well. For this reason, adaptivity in a dynamic setting is still required. This is similar to tracking a shifting optimum, which is what the majority of the literature on dynamic EAs is about [7].

A second advantage is that a strategy, assuming it is designed well, can be understood much better and allows for easier implementation and understanding in practice. It is harder to understand why a certain list of decisions is optimized than a strategy that describes exactly the conditions under which a certain decision is made. This builds an important bridge between computation and its practical use.

One disadvantage of the approach is that a strategy needs to be designed in addition to modeling the problem. Theoretically, a strategy can be built using genetic programming techniques and building functions like neural nets on the basis of all available variables, but typically the results are then again hard to interpret. Also, it takes considerable additional computational effort to build such a general strategy. Still, designing a proper strategy by hand can also be hard. The capacity of the function class that describes the strategy must be adequate. In other words, the strategy must be able to express a good way of making decisions. In addition, the design of the strategy strongly influences the smoothness of the adaptation required from the EA. The less adaptation is required, the less resources are required from the EA (such as population size) to obtain good results.

Finally, we note that whereas the use of a good strategy is expected to work well with EAs and scenario–based optimization, this is typically not the case for the combination of strategies with the expected–value approach. The reason is that in the expected–value approach, the EA on the one hand will attempt to build a strategy for the expected–value case, as this case is used in the approximation of future situations. However, the current situation is not an expected–value case, but a real sampling from the distribution. Hence, the EA will also attempt to tailor the strategy to this particular situation. Even worse, many strategies may fit the expected–value case and give exactly the same result on the expected–value case. However, these strategies may be totally useless for actual cases. Hence, much more computational resources are required for the EA to distinguish between good and bad strategies than if scenario–based optimization is used. Moreover, the strategy will not evolve to a stable situation that smoothly changes over time as the problem changes, which is undesirable.

An experimental verification on the pickup problem using a simple strategy is given in Figures 2, 3 and 4. The strategy computes, using maximum–likelihood estimates, the parameters of the distribution for the package locations. As time goes by, this estimation becomes more exact. As a free parameter, the strategy has a threshold. The output of the strategy is 1 (i.e. perform a pickup) if the package location has an estimated density of at least the value of this threshold and the action doesn't lead to a direct loss. The latter addition is required to prevent the vehicle to drive between clusters in the case of the four normal distributions for the package generating distribution. The strategy may not lead to optimal choices, but it is a sensible approach. The results in Figure 2 show that indeed, for the case of one normal distribution, using a strategy is slightly inferior to directly optimizing decisions, but still outperforms the hillclimber. In the case of four normal distributions, the approach is much better than directly optimizing decisions. Additional results in Figure 3 show that even for smaller population sizes for the EA, using a strategy in combination with scenarios still gives good results. Using a strategy in combination with the expected value scenario however only gives good results if the largest population size is used. Otherwise, the capacity for good optimization quickly fades. Figure 4 shows the difference in smoothness of the strategy parameter to be optimized in a typical run. Clearly, when using scenarios, the strategy parameter is much more stable, allowing good results even for smaller population sizes.

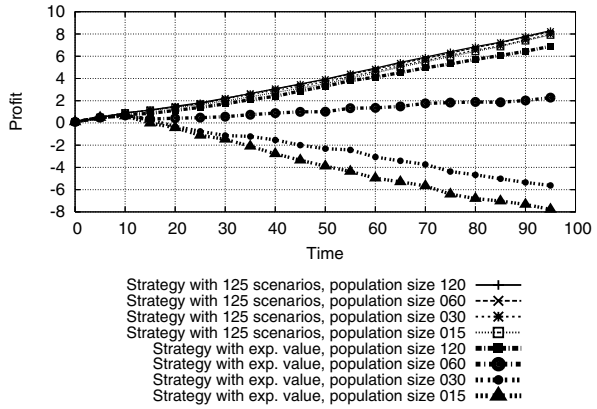

Figure 3: Results on the pickup problem with a single normal distribution for package generation, $N^{\text{future}} = 8$ and optimization using a strategy with 125 scenarios and a strategy with the expected value scenario; results are averaged over 50 runs.

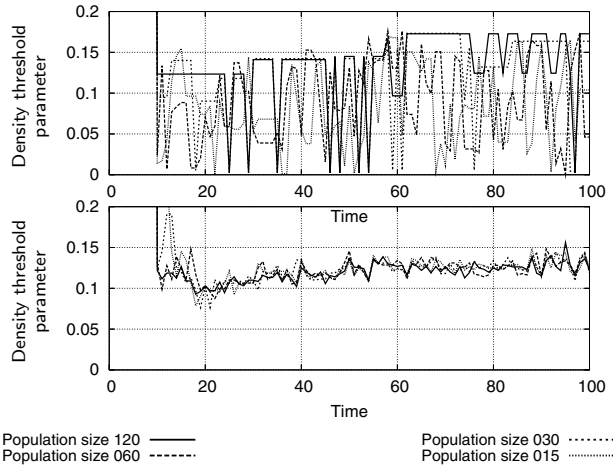

Figure 4: The strategy parameter when using the expected value (top) and 125 scenarios (bottom) in typical runs of the EA for various population sizes.

4. ALGORITHMIC FRAMEWORK

In Figure 5 pseudo–code is given that summarizes the approach proposed in the previous section. The general idea is that in addition to a population, a current best strategy is maintained. This allows the EA to be run continuously. Whenever a decision needs to be made, the current best strategy can be applied. Evaluation of a strategy is done by running that strategy in the simulator multiple times using different random seeds. The quality of a strategy is then measured by its average evaluation value. The variance is also stored. The variance is required to compare the best strategy in the population with the current best strategy. Because multiple random seeds are used, corresponding to multiple drawings from the probability distribution that underlies the problem, statistical hypothesis tests are required to be certain that an improvement has been obtained. The statistical hypothesis test that we used in our experiments is the Aspin–Welch–Satterthwaite (AWS) T–test at a significance level of $\alpha = 0.05$. The AWS T–test is a statistical hypothesis test for the equality of means in which the equality of variances is not assumed [13].

```
1   R ← InitializeRealWorld()
2   S ← InitializeSimulation()
3   P ← InitializePopulation()
4   s ← RandomScenarioSeeds()
5   for i ← 0 to |P| − 1 do
      5.1  EvaluateStrategy(P_i, s)
6   s^best ← SelectBestStrategy(P)
7   while t^now < t^end do
      7.1  EvolvePopulationOneGeneration()
      7.2  s ← RandomScenarioSeeds()
      7.3  for i ← 0 to |P| − 1 do
             7.3.1  EvaluateStrategy(P_i, s)
      7.4  EvaluateStrategy(s^best, s)
      7.5  s^candidate ← SelectBestStrategy(P)
      7.6  if SignificantlyBetter(s^candidate, s^best) then
             7.6.1  s^best ← s^candidate
```

```
EvaluateStrategy(strategy, s)
1   A ← ∅
2   for i ← 0 to N^scenarios − 1 do
      2.1  S' ← CloneSimulation(S)
      2.2  SetSimulationRandomSeed(S', s_i)
      2.3  v ← RunSimulation(S', strategy)
      2.4  A ← A ∪ {v}
3   ComputeAverageAndVariance(A)
```

Figure 5: Outline of algorithmic framework. EA (top) and evaluating a strategy (bottom).

Note that this framework is independent of the underlying dynamic EA and mainly describes the proper tackling of time–dependence. Implications of different systemic dynamics (e.g. rate and severity of changes) still play a key role in designing the underlying EA [7]. Here we specifically focus on time–dependence however. Also note that online learning and updating of the simulation are not presented above. These operations can be done simultaneously with the running of the EA (i.e. a separate thread) or in sequence (i.e. every so many generations).

5. EXPERIMENTS

5.1 Inventory management

Inventory management (IM) is an important area in logistics [16, 18]. In Figure 6 a schematic overview is given of IM problems. A general description of IM is the following. Buyers, also called customers and denoted C_i, order goods from a store. The number of goods and the frequency of ordering is called the demand and is denoted D_i for customer C_i. To prevent going out of stock, the store keeps an inventory. This inventory must however be replenished from time to time. Because the delivery of new stock from the store's suppliers also takes time (called the lead time, denoted L_j for supplier S_j), the replenishment–order must be placed before going out of stock. Time–dependence plays an important role here. The decision of whether or not to place a replenishment–order at a certain point in time has great future consequences because it determines future inventory levels. Also, a decision to place an order at a supplier leads to a future event (delivery of goods) that is a response to placing the order, making it impossible to completely sample a scenario prior to optimizing the decisions in that scenario. Maintaining a myopic view for IM problems leads to frequent out–of–stock or unnecessary–stock–surplus situations as future dynamic demand is not sufficiently taken into account. If this future dynamic demand is known, a non–myopic view enables to see how large the inventory should be in the (near) future. Hence, better profits can then be made as out–of–stock situations can be prevented.

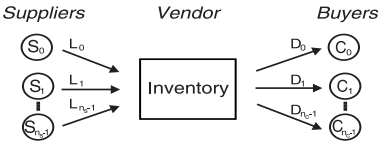

Figure 6: Overview of inventory management.

Many models exist for simple to hard problems in IM. For the simplest problems, exact optimal strategies are known. For practical problems however, there are typically multiple store–suppliers to choose from with different delivery times, order magnitudes and costs. Also demands and lead–times tend to be stochastic rather than deterministic. Although these aspects increase the benefit of a non–myopic view, they also make the problem harder to solve [16]. Consequently, no exact optimal strategies exist for the harder and more general cases. For specific cases, specific heuristics exist. There is no general flexible approach however that is applicable to a variety of IM problems. Here we show how, as a proof–of–principle, the framework in this paper may serve as such a general approach.

5.2 Experimental setup

We have designed two IM experiments. For both experiments, inventory is to be managed for 129600 minutes, i.e. 90 days. Orders can be placed any minute of the day.

5.2.1 Problems

Problem A represents problems of the type for which an optimal strategy can be computed beforehand. There is one supplier and one product. Product quantities are integer. The product is sold to the buyers at a price of 50 and bought from the supplier at a price of 20. A fixed setup cost for each order placed at a supplier is charged at a price of 50. Inventory holding costs are 1 per day per unit. The lead time of the supplier is fixed to 3 days. The demand is fixed to an order of 1 item every hour.

Problem B represents problems for which there is not a known optimal strategy. There are two suppliers. One supplier is cheaper than the other. The more expensive supplier can supply immediately, but costs twice as much. This type of setting is popular in IM research. It is typically known as IM with emergency replenishments and is known to be a hard problem [16]. The second supplier is used only if the stock has become really low and stock outs are imminent. To add to the difficulty of the problem, we have made the lead–time of the cheapest supplier both stochastic and periodically changing. The lead time of the slower supplier is normally distributed with mean (in minutes) of $4320\,(\cos((2\pi t)/43200)+1)\,/2$, i.e. it varies between 0 and 3 days and the period–length of the cosine is 30 days. The variance is $1440^2\,(\cos((2\pi t)/43200)+1)\,/2$, i.e. it varies between 0 and 1440 days, corresponding to a maximum standard deviation of 38 days with the same period–length as the mean. The periodically changing lead time causes the optimal strategy to change with time as well. Also, the demand is now stochastic. The time between two subsequent orders is normally distributed with a mean of one hour and a variance of 60 hours. The amount ordered is also normally distributed, with a mean of 3 and a variance of 9 products. For this setting, there aren't any known heuristics.

5.2.2 Dynamic EA

The EA follows the framework from in Section 4. The strategy that we employed is a common one in IM. The strategy is a so–called (s, Q) strategy [18]; s is called the re–order point and Q the order–up–to size. One such strategy is used for each supplier. Hence, the genotype contains $2n_s$ real values to be optimized, where n_s is the number of suppliers. If the stock drops below the re–order point s_i of supplier i, and no order is currently outstanding for supplier i, a new order is placed at supplier i of size $Q - stocklevel$. Thus, in the case of two suppliers, if an order from the cheaper supplier is running late, the stock level will drop further and the rule for the more expensive, emergency supplier becomes active. It is not known whether this strategy can be optimal for problem B, but it is an often used, sensible choice.

We used two different EA settings, one setting corresponds to a situation in which there is only very little time to do optimization and thus the EA resources are small. The other setting corresponds to a situation in which there is more time and thus the EA resources are larger. In the small settings, the population size is 50, scenario–evaluation simulates 5 days into the future, 5 generations of the EA can be done per day, and 10 scenarios are used. In the larger settings, the population size is 200, scenario–evaluation simulates 7 days into the future, 5 generations of the EA can be done per hour and 30 scenarios are used. To facilitate the simulation of the future, the EA learns the distribution behind the stochasticity of the buyer. The stochasticity of the supplier is assumed to be known.

5.3 Results

In Figure 7 the average profit obtained over 50 independent runs is shown for problems A and B for both EA settings. The EA approach is a scalable technique on both problems in the sense that allowing more resources results in better solutions. Investing in computing power thus results in a better policy for a vendor. Moreover, even the small settings for the EA lead to profits. The maximum profit that can be obtained on problem A is 59231. This profit corresponds to a setting of the strategy $((s, Q) = (72, 118))$ that is far outside the range in which we initialized the EA $((s, Q) \in [0, 25]^2)$. Out of all settings in the initialization range, the maximum profit is only 17050. The EA is thus also capable of finding much better solutions when initialization is suboptimal.

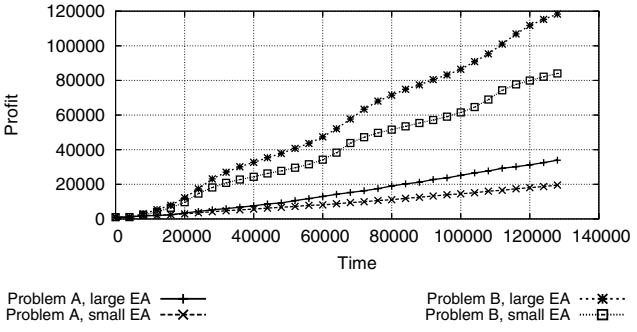

Figure 7: Results on inventory–management problems, averaged over 50 independent runs.

A second thing that stands out is that the profits on problem B are higher. The average demand in problem B is three times higher than in problem A. Indeed, the EA is able to obtain a profit of about 3 times higher than on problem A even though problem B is far more difficult.

Figure 8 shows the strategies obtained with the large EA settings for both problems in a typical run of the EA. The lack of stochasticity in problem A translates into finding a stable strategy by the EA very quickly and maintaining that strategy throughout the run. On problem B, the adaptive capacity of the EA allows the algorithm to continuously adapt the strategies and find better solutions for the situation at hand as the lead time of the cheapest supplier changes with time. The periodic change of the lead time of the cheapest supplier (S_0) is clearly translated into a periodic change in strategy. When the average and variance of the lead time of the cheapest supplier are small, less products need to be ordered and the threshold can be lower. The threshold for emergency replenishments can even become 0. When the lead time is the largest, emergency replenishments may become necessary and concordantly, the EA proposes a strategy in which emergency replenishments are made before the stock runs out completely. Also, in this case the re–order point for the cheapest supplier is much higher as is the number of products ordered from that supplier. It can be seen in Figure 9 that emergency replenishments are indeed made during the periods when the cheapest supplier is less reliable. Furthermore, note that the strategies are not exactly the same in each period. Note that while the EA is optimizing strategies, it is also still learning distributions. Learning converges to the true distribution over time. Finally, the periodic change in the lead time of the cheapest supplier can also be seen back in the obtained profits in Figure 7. When the lead time of the cheapest supplier is the smallest, the EA succeeds in finding a strategy that uses this supplier more and therefore obtains more profit, resulting in a steeper slope of the profit–versus–time graph at these moments.

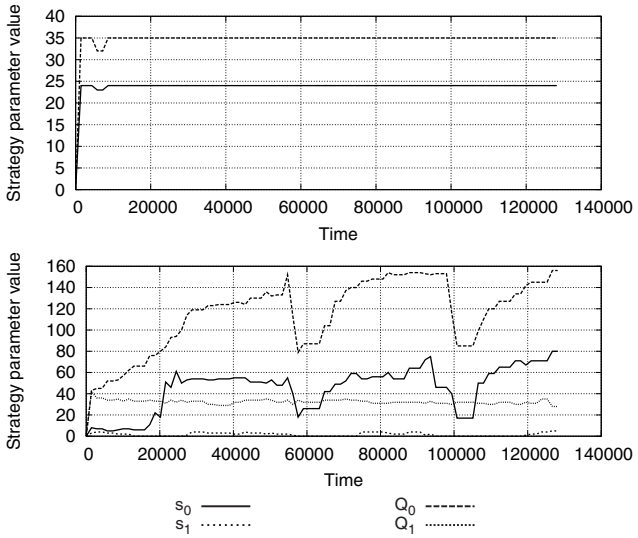

Figure 8: Strategies evolved in a typical run of the EA on problem A (top) and problem B (bottom).

Finally, we want to point out that in addition to the positive results of the anticipatory EA approach on the problems in this paper, one of the most beneficial aspects of this approach is that when the problems are changed (e.g. add more suppliers or buyers or change distribution settings), the approach remains the same and is able to obtain posi-

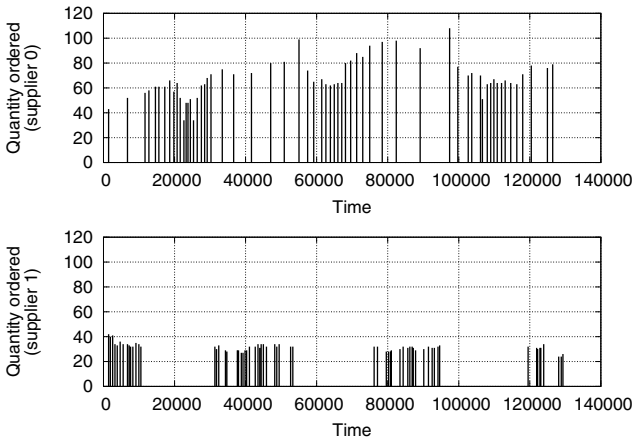

Figure 9: Quantities ordered in a typical run of the EA on problem B.

tive results. In other words, no in–depth redesigning needs to be done when something changes in the definition of the problem. Often, well–designed heuristics need to be completely redesigned when transferring a problem from theory into practice because of discrepancies between the theoretic case and the practical case and the fact that the heuristics are very problem–specific.

6. SUMMARY AND CONCLUSIONS

In this paper we have focused on designing evolutionary algorithms for solving online stochastic dynamic optimization problems. We have indicated that an EA is not only required to be able to track optima as they shift with time, an EA is also required to be able to perform anticipation, i.e. take into account consequences of decisions taken earlier. To this end, optimization needs to be performed for the current decision but also, simultaneously, for future decisions in future, anticipated situations.

We have argued that if the expected value is representative of the stochasticity in the problem, a list of decisions for the current situation and for future situations can be optimized. Evaluation of future situations is done by replacing the stochasticity in the simulation with the expected value. If the expected value is however not representative, as is the case in many practical applications, this approach fails to perform efficient optimization. In this case, a strategy can be evolved. Evaluation of the strategy in future, anticipated, situations is then done using multiple scenarios, i.e. simulations with a fixed random seed. We have shown that, if a strategy is designed properly, the optimal parameters for this strategy can be tracked using the adaptivity of EAs, thus eliminating the need for re–optimizations and allowing optimization to be performed continuously. We have also shown that the combination of strategies and the expected value case only works with relatively great computational effort and generally doesn't result in smoothly adapting trajectories of the strategy parameters. An additional advantage of the approach proposed in this paper is that a strategy can be interpreted more easily by the end–user, thereby giving more insight into the decision process compared to when decisions are optimized directly.

We have presented an algorithmic framework for this approach. This framework can be used in the future design of EAs for specific problems, which is one of the main avenues of future research that we shall explore.

7. REFERENCES

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.

[2] R. Bent and P. Van Hentenryck. Online stochastic and robust optimization. In M. J. Maher, editor, *Proceedings of the Asian Computing Science Conference — ASIAN–2004*, pages 286–300, Berlin, 2004. Springer–Verlag.

[3] R. Bent and P. Van Hentenryck. Regrets only! online stochastic optimization under time constraints. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the National Conf. on Artificial Intelligence — AAAI–2004*, pages 501–506, Menlo Park, California, 2004. AAAI Press.

[4] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge Univ. Press, 1998.

[5] P. A. N. Bosman. Learning, anticipation and time–deception in evolutionary online dynamic optimization. In S. Yang and J. Branke, editors, *Evolutionary Algorithms for Dynamic Optimization Problems EvoDOP Workshop at the Genetic and Evolutionary Computation Conference GECCO–2005*, pages 39–47, New York, New York, 2005. ACM Press.

[6] P. A. N. Bosman and H. La Poutré. Computationally intelligent online dynamic vehicle routing by explicit load prediction in an evolutionary algorithm. In T. P. Runarsson et al., editors, *Parallel Problem Solving from Nature — PPSN IX*, pages 312–321, Berlin, 2006. Springer Verlag.

[7] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, Norwell, Massachusetts, 2001.

[8] J. Branke and D. Mattfeld. Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research*, 43(15):3103–3129, 2005.

[9] M. R. Caputo. *Foundations of Dynamic Economic Analysis*. Cambridge Univ. Press, Cambridge, 2005.

[10] H. Chang, R. Givan, and E. Chong. On–line scheduling via sampling. In *Artificial Intelligence Planning and Scheduling (AIPS'00)*, pages 62–71, 2000.

[11] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learing*. Addison Wesley, Reading, Massachusetts, 1989.

[12] M. Grötschel, S.O. Krumke, and J. Rambau. *Online optimization of large scale systems*. Springer–Verlag, Berlin, 2001.

[13] M.G. Kendall and A. Stuart. *The Advanced Theory Of Statistics, Volume 2, Inference And Relationship*. Charles Griffin & Company Limited, 1967.

[14] A. Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Technical University of Denmark, Denmark, 2000.

[15] L. Mercier and P. Van Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic programs. In *Proceedings of the International Joint Conf. on Artificial Intelligence — IJCAI–2007*, 2007.

[16] S. Minner. Multiple–supplier inventory models in supply chain management: A review. *International J. of Production Economics*, 81–82:265–279, 2003.

[17] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. binary parameters. In A. E. Eiben et al., editors, *Parallel Problem Solving from Nature – PPSN V*, pages 178–187, Berlin, 1998. Springer–Verlag.

[18] S. Nahmias. *Production and Operations Analysis*. Irwin, Inc., Homewood, IL, 1997.

[19] M. Savelsbergh. DRIVE: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490, 1998.

[20] J. Sgall. On–line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 196–232. Springer–Verlag, Berlin, 1998.

[21] J. I. van Hemert and J. A. La Poutré. Dynamic routing problems with fruitful regions: models and evolutionary computation. In X. Yao et al., editors, *Parallel Problem Solving from Nature – PPSN VIII*, pages 692–701, Berlin, 2004. Springer Verlag.