Constraint-directed Search in Computational Finance and Economics

Edward Tsang

Centre for Computational Intelligence in Finance & Economics (CCFEA)
University of Essex, Colchester, UK
edward@essex.ac.uk

1 Use the force

Constraints shield solutions from a problem solver. However, in the hands of trained constraint problem solvers, the same constraints that create the problems in the first place can also guide problem solvers to solutions. Constraint satisfaction is all about learning how to flow with the force of the constraints.

Examples of using constraints to guide one's search are abundant in complete search methods (e.g. see [1, 2]). Lookahead algorithms propagate constraints in order to (a) reduce the remaining problem to smaller problems and (b) detect dead-ends. Dependency-directed backtracking algorithms use constraints to identify potential culprits in dead-ends. This helps the search to avoid examining (in vain) combinations of variables assignments that do not matter.

Constraint-directed search is used in stochastic search too. Constraints were used in Guided Local Search (GLS) [3] and Guided Genetic Algorithm (GGA) [4] to guide the search to promising areas of the search space. In stochastic methods, a constraint satisfaction problem is handled as an optimization problem, where the goal is to minimize the number of constraints violated. The approach in GLS is to use constraints to augment the objective function. This helps local search to escape local optima. GGA uses the GLS penalty scheme to change the behaviour of genetic algorithms. This results in a more robust algorithm which finds quality results consistently. GLS and GGA have been applied to many optimization problems, including the well-known travelling salesman problem and quadric assignment problem.

The GLS idea was generalized to "penalties" and "incentives" in evolutionary computation. This paper explains how such ideas were applied to two applications in finance and economics: financial forecasting and automated bargaining.

2 Constraints in financial forecasting

In forecasting, the goal is to predict the value of a variable, which defines the target. The challenge in forecasting is (i) to find a set of variables, and (ii) to find a function that maps these variables to the target. There is no limit in the format of this function. It can be a mathematical function. It can also be a program procedure.

There is no guarantee that such functions exist. If they do, then finding the relevant variables is essential for their discovery. EDDIE (which stands for Evolutionary Dynamic Data Investment Evaluator) is a framework for finding such functions [5]. Experts channel their financial expertise into the system through the suggestion of variables. EDDIE attempts to find functions that map these variables to the target.

EDDIE attempts to predict a particular form of patterns: whether prices will go up by r% or more within the next n days. (Here r could be a negative number). In that case, the target can be represented by a Boolean variable T. If T is true, it means prices will go up by r% or more within the next n days, which represents an investment opportunity. For example, domain experts may suggest that the current price, the "50 days moving average" and "volatility" (which could be measured by the normalized standard deviation of the previous, say, 25 days) are indicators of the future price. An example of a function is:

```
IF the current price is 6.24% above the 50 days moving average
   THEN IF volatility is above 1.85,
             THEN T is True;
          ELSE T is False;
ELSE IF the current price is 12.49% below the 50 days moving average
          THEN T is False;
       ELSE T is True;
```

In this example, the function is represented by a tree. EDDIE is responsible for finding the structure of the tree, as well as the thresholds such as 6.24%, 1.85.

The search for variables is crucial to the success of forecasting. This is the job of the finance experts, which will not be discussed here. (This job can be helped by EDDIE, see [6]). Faced by EDDIE is a huge search space of tree structures and thresholds. EDDIE searches the space with genetic programming. Pretty standard genetic programming techniques were adopted, except the use of constraints, which is described below.

In EDDIE, precision refers to the percentage of "True" predictions that turn out to be correct in reality. Recall refers to the percentage of investment opportunities that were correctly predicted "True" by EDDIE. Failure in picking an opportunity is not as serious as making a wrong decision to invest, because the latter could lead to losses. That means precision is more important than recall in financial forecasting. Having said that, if a forecasting tool fails to pick up any investment opportunities at all (i.e. recall=0), then this tool is useless. Therefore, one would like to have a handle to balance between precision and recall. This is attempted by FGP2, a version of EDDIE. Following is a brief summary of FGP2; details can be found in [7].

FGP2 aims to concentrate the search on areas of the space where trees have higher precision. To achieve that, FGP2 augmented the objective function with a constraint. The augmented objective function encouraged trees that make a certain percentage of their predictions "True". If the percentage of "True" predictions by a tree is not within a range constrained by the user, its fitness is significantly reduced. This range constraint is set by the user; it typically reflects conditions of the current market.

Trees capture patterns in the data. The EDDIE experience was that, even with variables drawn from text-books (namely technical trading indicators), patterns could be found in some of the stocks [7]. With variables of better quality, patterns with extremely high precision could be found [8]. Patterns do not appear in all stocks. Besides, the market is changing very fast in recent years (with the significant growth of algorithmic trading), which hinders learning. Nevertheless, one does not have to find all patterns to benefit from forecasting. A single opportunity, if detected, could provide a trader with valuable reward. Whenever patterns exist, having a forecasting tool like EDDIE is better than not.

3 Constraints in automated bargaining

Game theory is often used in a political or military context to explain conflicts between countries. More recently it has been used to map trends in the business world, ranging from how cartels set prices to how companies can better sell their goods and services in new markets. It has become an important area in economics, for which Nobel Prizes have been awarded (e.g. Aumann and Schelling in 2005).

Bargaining is a main subject in game theory. One of the fundamental bargaining models was Rubinstein's 1982 model. Under this model, two players bargain to share a pie. They make alternative offers. For example, the first player may offer to take 65% of the pie. The second player may either accept it or reject the 35% offerred. If he rejects this offer, he will have to give a counter offer, e.g. he may ask for 51%. However, both players' utilities drop exponentially over time. That means getting 51% in the second round may not worth as much as accepting 35% in the first round. This motivates both players to accept an offer as soon as possible. It is worth pointing out that the two players may have different utility discount rates. The value of the discount rate determines their bargaining power. A player with a higher discount rate is in a weaker position to bargain.

A players optimal strategy depends on the other players strategy. Subgame equilibrium is the optimal strategy by both players, given their belief of their opponents strategies. To derive the subgame equilibrium, Rubinstein assumed complete information, i.e. each player knows both discount rates, and know that the opponent knows such. Rubinstein also assumed perfect rationality by both players [9]. Subgame equilibrium was derived recursively by Rubinstein: to calculate the first players optimal strategy, one has to solve the subproblem of the second players strategy. This in turn can be calculated by the first players optimal strategy in the third round should the second player make a counter offer in the previous round. The subproblems can be solved recursively till both players utilities drop to a fix point.

In game theory, subgame equilibrium is typically derived mathematically. There are two serious drawbacks in this approach. Firstly, it assumes perfect rationality in decision making. In practice, decision making often involves computation (chess is a good example). Therefore, computational intelligence determines the effective rationality (I call this the CIDER theory, see [10]). Secondly, mathematical derivation of subgame equilibrium is laborious. A slight change of the bargaining model (for example, when a player has an outside option which guarantees him, say, 36% of the pie) would typically require complete revision of the derivation.

The above drawbacks of the mathematical approach motivate a co-evolutionary approach, where each of the two players is modelled by a population of strategies [11]. A strategys fitness is evaluated through playing it with strategies by the opponent. In evolutionary computation, a strategy's chance of survival depends on its fitness. That means under this approach, the perfect rationality assumption is replaced by reinforcement learning, which is closer to reality. Besides, this approach is robust: it can easily cope with slight changes to the bargaining model. It can easily capture asymmetric information or asymmetric ability by the two players.

Jin et al used genetic programming to approximate subgame equilibrium [11]. Bidding strategies were represented by functions. Under this approach, each player searches in the space of functions. Unfortunately, the search space is huge. Besides, only a very small proportion of the functions in the search space are sensible. For example, a random strategy would typically return a bid of below 0% or above 100% of the pie. Standard genetic programming failed to find sensible strategies consistently.

Following EDDIE's experience, Jin and Tsang used constraints to guide the search. To do so, desirable attributes were identified for bidding strategies. Firstly, a strategy should return a value between 0 and 1. Secondly, the value that a bidding strategy returns should ideally be inversely proportional to the player's own utility discount rate. Thirdly, the value that a strategy returns should ideally be proportional to the opponent's discount rate. These desirable attributes were translated into incentives, which augmented the objective function.

With the help of incentives, the majority of the populations contained usable bidding strategies (which demand a value between 0% and 100%). The subgame equilibrium found by co-evolution was very close to the theoretical solutions in Robinsteins 1982 bargaining model. With minor modifications, the programs were applicable to variations of Rubinstein's bargaining model [9]. For these simple bargaining models, the subgame equilibrium found by co-evolution approximated the theoretical solutions. These results suggest that constraint-directed co-evolutionary is a useful approach to approximate subgame equilibrium in bargaining.

References

1. Tsang, E.P.K.: Foundations of constraint satisfaction. Academic Press, London and San Diego. (1993)

2. Rossi, F., van Beek, P., Walsh, T. (ed.): Handbook of Constraint Programming, Elsevier. (2006)
3. Voudouris, C., Tsang, E.P.K.: Guided local search. Handbook of metaheuristics, Glover, F.(eds), Kluwer. 185–218 (2003)
4. Lau, T.L., Tsang, E.P.K.: Guided genetic algorithm and its application to radio link frequency assignment problems. Constraints, 6(4), 373–398 (2001)
5. Tsang, E.P.K., Yung, P., Li, J.: EDDIE-Automation, a decision support tool for financial forecasting. Journal of Decision Support Systems, Special Issue on Data Mining for Financial Decision Making. 37(4), 559–565 (2004)
6. Kampouridis, M., Tsang, E.: EDDIE for Investment Opportunities Forecasting: Extending the Search Space of the GP. In Proceedings of the IEEE Congress on Evolutionary Computation 2010, Barcelona, Spain. (to appear)
7. Li, J., Tsang, E.P.K.: Investment decision making using FGP: a case study. Proceedings, Congress on Evolutionary Computation, Washington DC, USA. 1253–1259 (1999)
8. Tsang, E.P.K., Markose, S., Er, H.: Chance discovery in stock index option and future arbitrage. New Mathematics and Natural Computation, World Scientific. 1(3), 435–447 (2005)
9. Rubinstein, A.: Perfect Equilibrium in a Bargaining Model. Econometrica. 50, 97–110 (1982)
10. Tsang, E.P.K.: Computational intelligence determines effective rationality. International Journal on Automation and Control. 5(1), 63–66 (2008)
11. Jin, N., Tsang, E.P.K., Li, J.: A constraint-guided method with evolutionary algorithms for economic problems. Applied Soft Computing. 9(3), 924–935 (2009)
12. Tsang, E.P.K.: Forecasting where computational intelligence meets the stock market. Frontiers of Computer Science in China, Springer. 53–63 (2009)