ДОДАТОК А

Текст програми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

"ЗАТВЕРДЖУЮ"
керівник атестаційної роботи
проф. Д.Е.Ситніков

РОЗРОБКА МЕТОДУ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ РИНКІВ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ КОМБІНАЦІЇ ТЕХНІЧНИХ ІНДИКАТОРІВ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

ГЮИК. 506490-01 12 01-ЛЗ

УЗГОДЖЕНО:

РОЗРОБИВ:

Ст.гр. ІТПм-18-1
К.Д.Карчевський

2019

ЗАТВЕРДЖЕНО

ГЮІК. 506490-01 12 01-ЛЗ

РОЗРОБКА МЕТОДУ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ РИНКІВ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ КОМБІНАЦІЇ ТЕХНІЧНИХ ІНДИКАТОРІВ

Текст програми

ГЮИК. 506490-01 12 01

Аркушів 12

2019

```
     analysis.test-algo.go

package analysis

import "math"

type testAlgorithm struct {
     atr
     macd
     trigger        func(*TriggerParams)
     Multiplier     float64
     MACDMultiplier float64
     lastBelowBand   float64
     lastAboveBand  float64
     isUp           bool
     comparePeeks   bool
}

func NewTestAlgorithm(size int, multiplier, macdMultiplier
float64) *testAlgorithm {
     return &testAlgorithm{
          atr:            *NewAtr(size),
          macd:           *NewMACD(12, 22, 9),
          Multiplier:     multiplier,
          MACDMultiplier: macdMultiplier,
          trigger:        func(*TriggerParams) {},
     }
}

func (s *testAlgorithm) Analyse(preData []*Candle, stream chan
*Candle) {
}

func (s *testAlgorithm) SetTrigHandler(handler
func(*TriggerParams)) {
     s.trigger = handler
}

func (s *testAlgorithm) LastTrends(data []*Candle) interface{}
{
     resultSize := len(data)
     aboveResult := make([]float64, resultSize)
     belowResult := make([]float64, resultSize)
     aboveResult[0] = data[0].High
     belowResult[0] = data[0].Low

     s.lastAboveBand = aboveResult[0]

     s.atr.iteration(data[0])
     s.macd.iteration(data[0])
     for i := 1; i < len(data); i++ {
          candle := data[i]
```

```
          preCandle := data[i-1]

          atr := s.atr.iteration(candle) * s.Multiplier
          signal, macd := s.macd.iteration(candle)

          macdDifference := (macd - signal) * s.MACDMultiplier

          belowBand := s.belowBand(candle, preCandle, atr,
macdDifference)
          aboveBand := s.aboveBand(candle, preCandle, atr,
macdDifference)
          direction := s.calculateTrendDirection(candle)

          if direction != s.isUp {
               s.isUp = direction
               s.trigger(&TriggerParams{candle,
s.getOrderPrice(candle), TrendDirection(direction)})
          }

          s.lastAboveBand = aboveBand
          s.lastBelowBand = belowBand
          aboveResult[i] = aboveBand
          belowResult[i] = belowBand
     }
     return [][]float64{belowResult, aboveResult}
}

func (s testAlgorithm) belowBand(candle, prevCandle *Candle,
atr, macd float64) float64 {
     belowBand := candle.Average() - atr - macd
     if s.lastBelowBand <= s.getComparableLow(prevCandle) {
          return Max(belowBand, s.lastBelowBand)
     }
     return belowBand
}

func (s testAlgorithm) aboveBand(candle, prevCandle *Candle,
atr, macd float64) float64 {
     aboveBand := candle.Average() + atr - macd
     if s.lastAboveBand >= s.getComparableHigh(prevCandle) {
          return Min(aboveBand, s.lastAboveBand)
     }
     return aboveBand
}

func (s testAlgorithm) getComparableHigh(candle *Candle)
float64 {
     if s.comparePeeks {
          return candle.High
     }
     return candle.Close
}
```

```
func (s testAlgorithm) getComparableLow(candle *Candle) float64
{
     if s.comparePeeks {
          return candle.Low
     }
     return candle.Close
}

func (s testAlgorithm) getOrderPrice(candle *Candle) float64 {
     if s.isUp {
          return math.Min(s.lastAboveBand, candle.High)
     }
     return math.Max(s.lastBelowBand, candle.Low)
}

func (s testAlgorithm) calculateTrendDirection(candle *Candle)
bool {
     if !s.isUp && candle.High > s.lastAboveBand {
          return true
     } else if s.isUp && candle.Low < s.lastBelowBand {
          return false
     } else {
          return s.isUp
     }
}
```

analysis/atr.go

```
package analysis

import "math"

type atr struct {
     Size      int
     prevClose float64
     trigger   func(params *TriggerParams)
}

func NewAtr(size int) *atr {
     return &atr{Size: size, trigger: func(*TP) {}}
}

func (a *atr) Analyse(preData []*Candle, stream chan *Candle) {

}

func (a *atr) SetTrigHandler(handler func(*TriggerParams)) {
     a.trigger = handler
}

func (a *atr) LastTrends(data []*Candle) interface{} {
```

```
	size := a.Size
	if len(data) < size {
		return make([]float64, 0)
	}

	result := make([]float64, len(data))
	prevClose := data[0].Close
	for i := 0; i < size; i++ {
		v := data[i]
		result[i] = getTR(v.High, v.Low, prevClose)
		prevClose = v.Close
	}
	result[size-1] = Average(result[:size]...)

	for i := size; i < len(data); i++ {
		v := data[i]
		tr := getTR(v.High, v.Low, prevClose)
		result[i] = getATR(tr, result[i-1], float64(size))
		prevClose = v.Close
	}
	a.prevClose = prevClose
	return [][]float64{result}
}

func getTR(high, low, prevClose float64) float64 {
	maxOfPrevious := math.Max(math.Abs(high-prevClose),
math.Abs(low-prevClose))
	return math.Max(high-low, maxOfPrevious)
}

func getATR(tr, lastAtr float64, size float64) float64 {
	return (lastAtr*(size-1) + tr) / size
}

func (a *atr) iteration(v *Candle) float64 {
	prevClose := a.prevClose
	high := v.High
	low := v.Low
	maxOfPrevious := math.Max(math.Abs(high-prevClose),
math.Abs(low-prevClose))
	tr := math.Max(high-low, maxOfPrevious)
	a.prevClose = v.Close
	return tr
}
```

analysis/macd.go

```
package analysis

import "math"

type macd struct {
```

```
	short   movingAverage
	long    movingAverage
	alias   movingAverage
	last    float64
	isUp    bool
	trigger func(params *TriggerParams)
}

func NewMACD(short, long, alias int) *macd {
	return &macd{short: NewEMA(short), long: NewEMA(long),
alias: NewEMA(alias), trigger: func(*TriggerParams) {}}
}

// TODO need to test
func (m *macd) Analyse(preData []*Candle, stream chan *Candle)
{
	m.LastTrends(preData)
	for v := range stream {
		if v.Final {
			m.iteration(v)
		} else {
			m.iterationImmutable(v)
		}
	}
}

func (m *macd) SetTrigHandler(handler func(*TriggerParams)) {
	m.trigger = handler
}

func (m *macd) LastTrends(data []*Candle) interface{} {
	macdArr, signalArr := make([]float64, len(data)),
make([]float64, len(data))
	if len(data) <= m.long.Size() {
		return nil
	}

	for i := 1; i < len(data); i++ {
		r, s := m.iteration(data[i])
		macdArr[i], signalArr[i] = r, s
	}
	return [][]float64{macdArr, signalArr}
}

func (m *macd) iteration(data *Candle) (float64, float64) {
	long, short, alias := m.long, m.short, m.alias

	val := data.Close
	long.Add(val)
	short.Add(val)
	result := short.Result() - long.Result()
	alias.Add(result)
```

```
	signal := alias.Result()
	if m.isUp && result < signal || !m.isUp && result > signal
{
		m.uTurn(data)
	}
	return result, signal
}

func (m *macd) iterationImmutable(data *Candle) (float64,
float64) {
	long, short, alias := m.long, m.short, m.alias

	val := data.Close
	result := short.ResultImmutable(val) -
long.ResultImmutable(val)
	signal := alias.ResultImmutable(result)
	difference := result - signal
	if (m.isUp && result < signal || !m.isUp && result >
signal) && math.Abs(difference) > 0.00015*data.Close {
		m.uTurn(data)
	}
	return result, signal
}

func (m *macd) uTurn(candle *Candle) {
	m.isUp = !m.isUp
	m.trigger(&TriggerParams{Candle: candle, Direction:
TrendDirection(m.isUp), Value:candle.Close})
}
```

analysis/psar.go

```
package analysis

import (
	"fmt"
	"log"
	"math"
	"sync"
)

type psar struct {
	isUp                  bool
	acc, MinAcc, MaxAcc   float64
	lastSAR, extremeVal   float64
	curCandle, preCandle  *Candle
	sarHandler            func(*TriggerParams)
	stream                *chan *Candle
	wg                    sync.WaitGroup
}

func (tr *psar) SetOnTrig(handler func(*TriggerParams)) {
```

```
	tr.sarHandler = handler
}

func (tr *psar) sar() float64 {
	return tr.lastSAR + tr.acc*(tr.extremeVal-tr.lastSAR)
}

func (tr *psar) Algorithm(data *Candle) float64 {
	var result float64
	high, low := data.High, data.Low

	if tr.isUp && tr.extremeVal < tr.curCandle.High {
		tr.acc = math.Min(tr.MinAcc+tr.acc, tr.MaxAcc)
		tr.extremeVal = tr.curCandle.High
	}
	if !tr.isUp && tr.extremeVal > tr.curCandle.Low {
		tr.acc = math.Min(tr.MinAcc+tr.acc, tr.MaxAcc)
		tr.extremeVal = tr.curCandle.Low
	}

	if tr.isUp {
		result = Min(tr.sar(), tr.curCandle.Low)

		if result > low {
			result = tr.extremeVal
			tr.uTurn(data)
			tr.extremeVal = low
		}

	} else {
		result = Max(tr.sar(), tr.curCandle.High)
		if result < high {
			result = tr.extremeVal
			tr.uTurn(data)
			tr.extremeVal = high
		}
	}

	return result
}

func (tr *psar) checkUTurn(data *Candle) {
	high, low := data.High, data.Low

	if tr.isUp {

		if tr.lastSAR > low {
			tr.uTurn(data)
			tr.extremeVal = low
		}
	} else {
```

```
          if tr.lastSAR < high {
               tr.uTurn(data)
               tr.extremeVal = high
          }
     }
}

func (tr *psar) uTurn(candle *Candle) {
     tr.isUp = !tr.isUp
     tr.acc = tr.MinAcc

     tr.sarHandler(&TriggerParams{candle, tr.lastSAR,
TrendDirection(tr.isUp)})
     tr.lastSAR = tr.extremeVal
}

func (tr *psar) LastTrends(data []*Candle) interface{} {
     l := len(data)
     res := make([]float64, l)
     res[0] = data[0].High
     low, high := data[0].Low, data[0].High

     if tr.lastSAR == 0 {
          if tr.isUp {
               tr.lastSAR = low
               tr.extremeVal = high
          } else {
               tr.lastSAR = high
               tr.extremeVal = low
          }
     }

     for i := 2; i < l; i++ {
          tr.curCandle = data[i-1]
          tr.preCandle = data[i-2]
          tr.lastSAR = tr.Algorithm(data[i])
          res[i] = tr.lastSAR
     }
     tr.wg.Wait()
     res1 := make([][]float64, 1)
     res1[0] = res
     return res1
}

func (tr *psar) Analyse(preData []*Candle, stream chan *Candle)
{
     tr.stream = &stream
     tr.LastTrends(preData)
     l := len(preData)
     tr.curCandle = preData[l-1]
     tr.preCandle = preData[l-2]
     tr.streamFunc(stream, tr.Algorithm)
```

```
}

func (tr *psar) streamFunc(stream chan *Candle, handler
func(candle *Candle) float64) {
     log.Println("starting stream on PSAR analyser")
     for v := range stream {
          if v.Final {
               tr.lastSAR = handler(v)
               tr.preCandle = tr.curCandle
               tr.curCandle = v
          } else {
               tr.checkUTurn(v)
          }
     }
}

func (tr *psar) StopAnalyse() {
     close(*tr.stream)
     fmt.Println("analyse stopped")
}
```

analysis/ma.go

```
package analysis

type MovingAverage interface {
     Add(val float64)
     Result() float64
     Size() int
}
type SMA struct {
     size     int
     isFilled bool
     lastInd  int
     data     []float64
     last     float64
}

func (ma *SMA) Size() int {
     return ma.size
}

func (ma *SMA) Add(val float64) {
     ma.data[ma.lastInd] = val
     ma.lastInd = (ma.lastInd + 1) % ma.size
     if ma.lastInd == 0 {
          ma.isFilled = true
     }
}

func NewSMA(size int) *SMA {
     return &SMA{size: size, data: make([]float64, size),
```

```
isFilled: false}
}

func (ma *SMA) Result() float64 {
     var siz = ma.size
     if !ma.isFilled {
          siz = ma.lastInd
     }
     return Average(ma.data[:siz]...)
}
     analysis/ma_test.go

     package analysis

     import "testing"

     func TestMovingAverage_Result(t *testing.T) {
          ma := NewSMA(5)
          ma.Add(4)
          ma.Add(1)
          ma.Add(1)
          ma.Add(1)
          ma.Add(1)
          ma.Add(2)
          if ma.Result() != 1.2 {
               t.Error("Ma value isn't correct: ", ma.Result(),
     ", data: ", ma.data)
          }
     }

     observer/statistics.go

package observer

import (
     "analysis"
     "exchange"
     "fmt"
     "math"
)

type StatisticParams struct {
     Side   service.OrderSide `json:"side"`
     Candle *analysis.Candle  `json:"candle"`
     Price  float64           `json:"price"`
     Qty    float64           `json:"qty"`
     Tax    float64           `json:"tax"`
}

func (s StatisticParams) String() string {
     return fmt.Sprint(s.Side, "\t", s.Qty, " ", s.Price, "\t")
}
```

```
type OrderStatistic struct {
     Volume          float64               `json:"volume"`
     OrderCount      int                   `json:"orderCount"`
     Profit          float64               `json:"profit"`
     MaxProfit       float64               `json:"maxProfit"`
     ProfitAfterTax  float64
`json:"profitAfterTax"`
     Drawdown        float64               `json:"drawdown"`
     TriggeredParams []StatisticParams
`json:"triggeredParams"`
     OnUpdate        func(StatisticParams) `json:"-"`
     calculator      ProfitCalculator      `json:"-"`
}

func (o *OrderStatistic) ProfitToDrawdown() float64 {
     if o.Drawdown == 0 {
          return math.Inf(int(1))
     }
     return o.Profit / o.Drawdown
}

func (o *OrderStatistic) Add(params StatisticParams) error {
     orderSize := params.Qty * params.Price
     o.TriggeredParams = append(o.TriggeredParams, params)
     if err := o.calculator.Add(orderSize, params.Side); err !=
nil {
          return err
     }

     o.Profit = o.calculator.profit
     o.Drawdown = math.Min(o.Profit, o.Drawdown)
     o.Volume += orderSize
     o.MaxProfit = math.Max(o.Profit, o.MaxProfit)
     o.ProfitAfterTax = o.Profit - o.Volume*params.Tax
     o.OrderCount++
     return nil
}

func (o *OrderStatistic) String() string {
     return fmt.Sprint("Profit: ", o.Profit, "\tDrawdown: ",
o.Drawdown, "\t P:DD: ", o.ProfitToDrawdown(),
          "\tOrderCount: ", o.OrderCount, "\t MaxProfit: ",
o.MaxProfit, "\t Volume: ", o.Volume, "\n", o.TriggeredParams)
}

type ProfitCalculator struct {
     profit     float64
     difference float64
     firstBuy   bool
     touched    bool
```

```
}

func (p *ProfitCalculator) SetProfit(profit float64) {
     p.profit = profit
     p.touched = true
}

func (p *ProfitCalculator) Add(vol float64, side
service.OrderSide) error {
     switch side {
     case service.SideBuy:
          p.firstBuy = p.firstBuy || !p.touched
          p.difference -= vol
          if !p.firstBuy && p.touched {
               p.profit = p.difference
          }

     case service.SideSell:
          p.difference += vol
          if p.firstBuy && p.touched {
               p.profit = p.difference
          }

     default:
          return fmt.Errorf("unknown order side: %s", side)
     }
     p.touched = true
     return nil
}
```

ДОДАТОК Б

Перелік графічного матеріалу

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

"ЗАТВЕРДЖУЮ"
керівник кваліфікаційної роботи
_____ проф. Д. Е. Ситніков

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ТОРГІВЛІ КРИПТОВАЛЮТАМИ

Перелік графічного матеріалу
ЛИСТ ЗАТВЕРДЖЕННЯ
ГЮИК.506440.002 ИЗ – ЛЗ

УЗГОДЖЕНО:

РОЗРОБИВ:
Ст.гр. ІТПм-18-1
К.Д. Карчевський

2019

Затверджую

ГЮИК.506440.002 ИЗ – ЛЗ

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ТОРГІВЛІ КРИПТОВАЛЮТАМИ

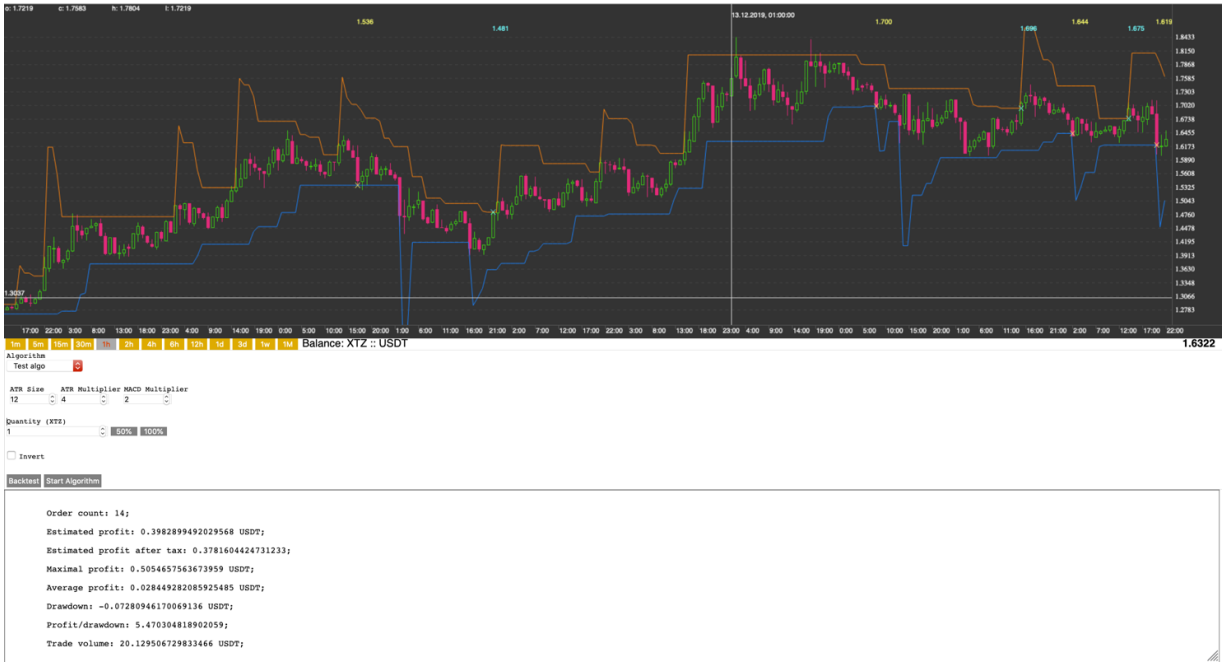
Перелік графічного матеріалу

ГЮИК.506440.002 ИЗ

Аркушів 9

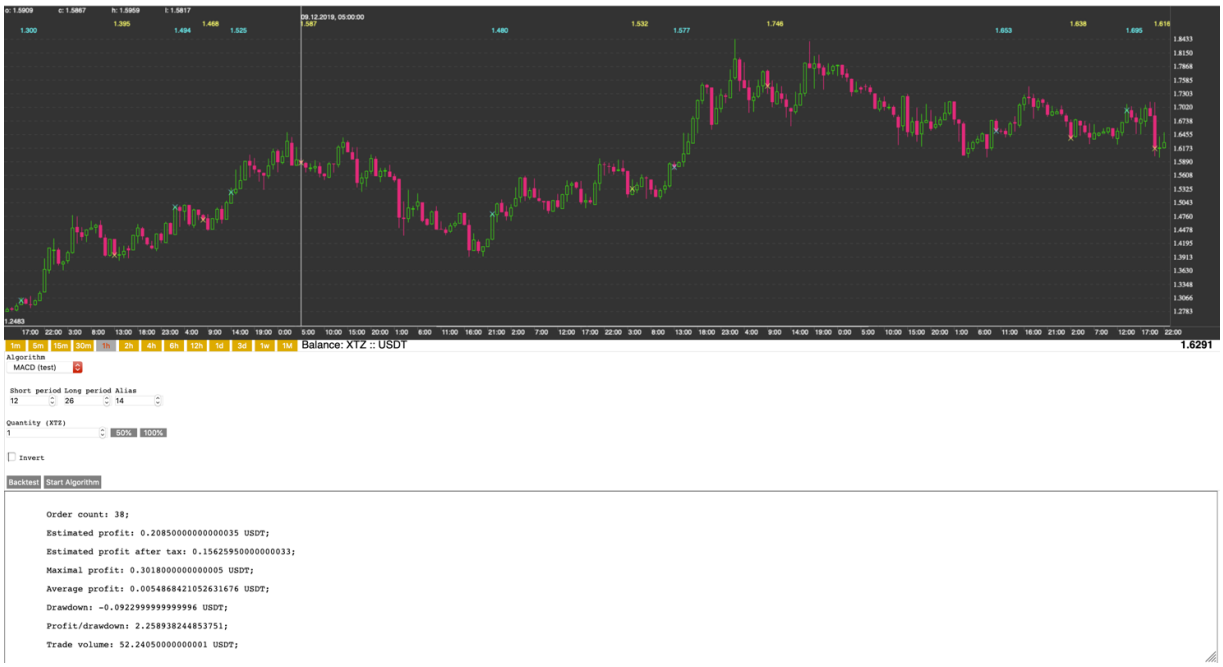
2019

РОБОТА ТА РЕЗУЛЬТАТ РОЗРОБЛЕНОГО ІНДИКАТОРА НА ГОДИННОМУ ІНТЕРВАЛІ



Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА MACD НА ГОДИННОМУ ІНТЕРВАЛІ



Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА PSAR НА ГОДИННОМУ ІНТЕРВАЛІ

Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ РОЗРОБЛЕНОГО ІНДИКАТОРА НА ШЕСТИГОДИННОМУ ІНТЕРВАЛІ

Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА MACD НА ШЕСТИГОДИННОМУ ІНТЕРВАЛІ

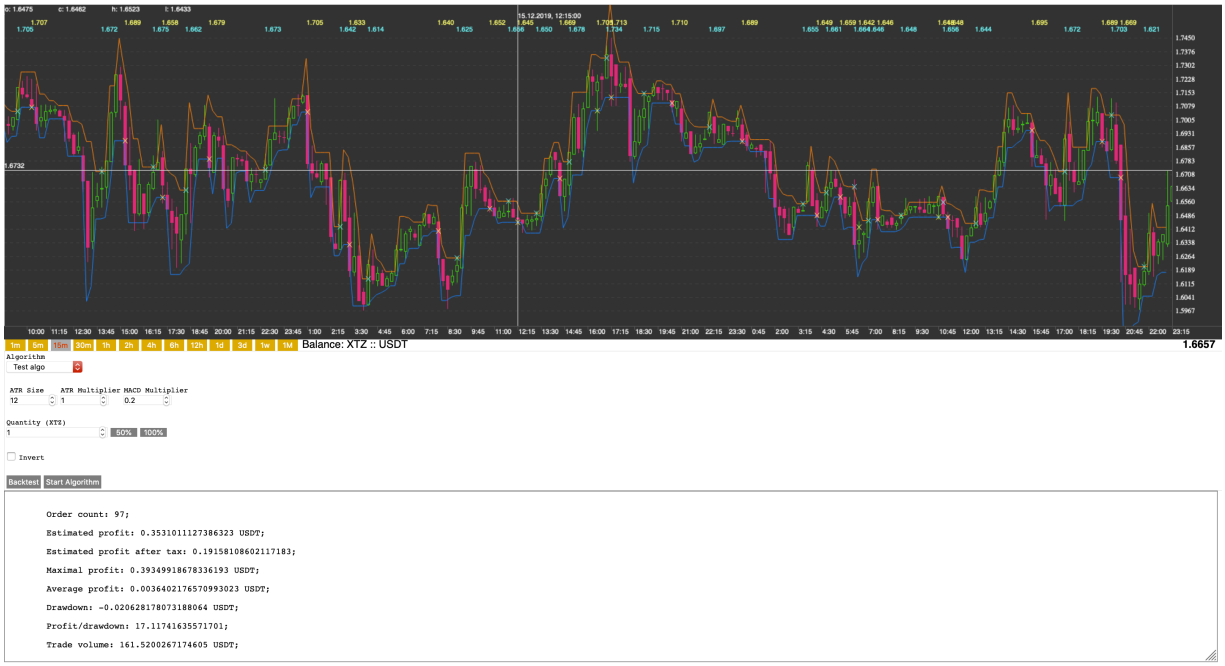
Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА PSAR НА ШЕСТИГОДИННОМУ ІНТЕРВАЛІ


```
Order count: 48;
Estimated profit: 0.5266 USDT;
Estimated profit after tax: 0.47406899999999996;
Maximal profit: 0.6151 USDT;
Average profit: 0.010970833333333332 USDT;
Drawdown: -0.07130000000000025 USDT;
Profit/drawdown: 7.385694249649342;
Trade volume: 52.531000000000006 USDT;
```

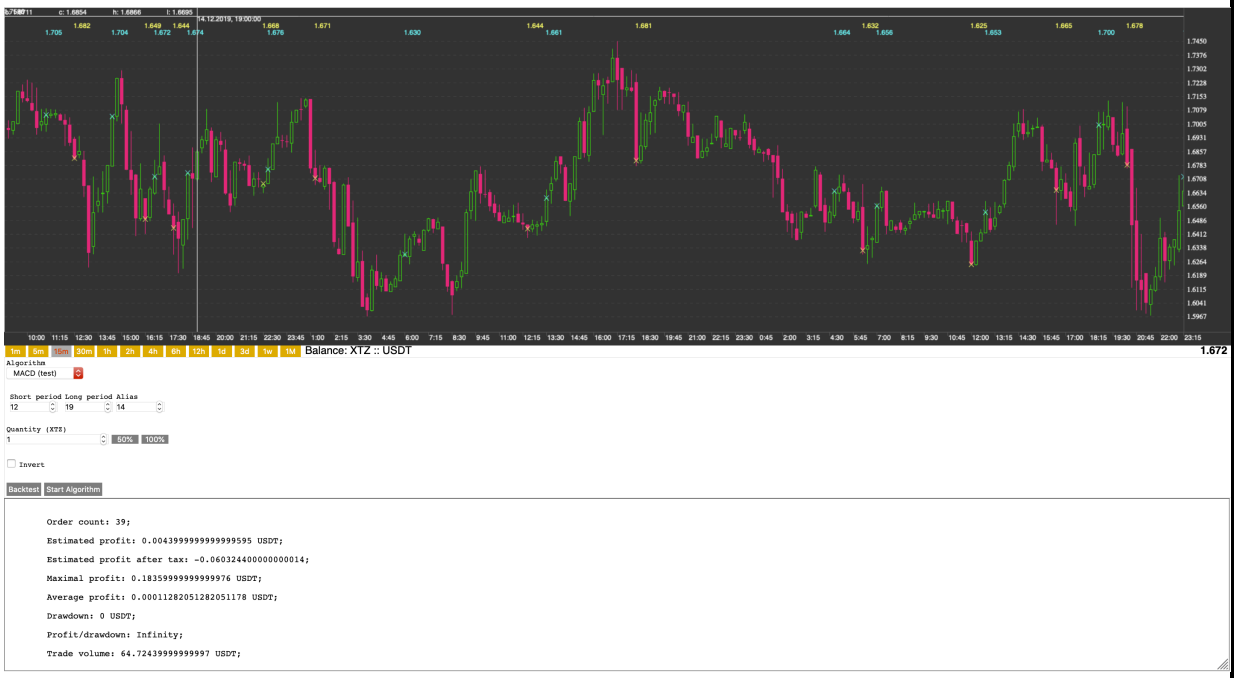
Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ РОЗРОБЛЕНОГО ІНДИКАТОРА НА П’ЯТНАДЦЯТИХВИЛИННОМУ ІНТЕРВАЛІ



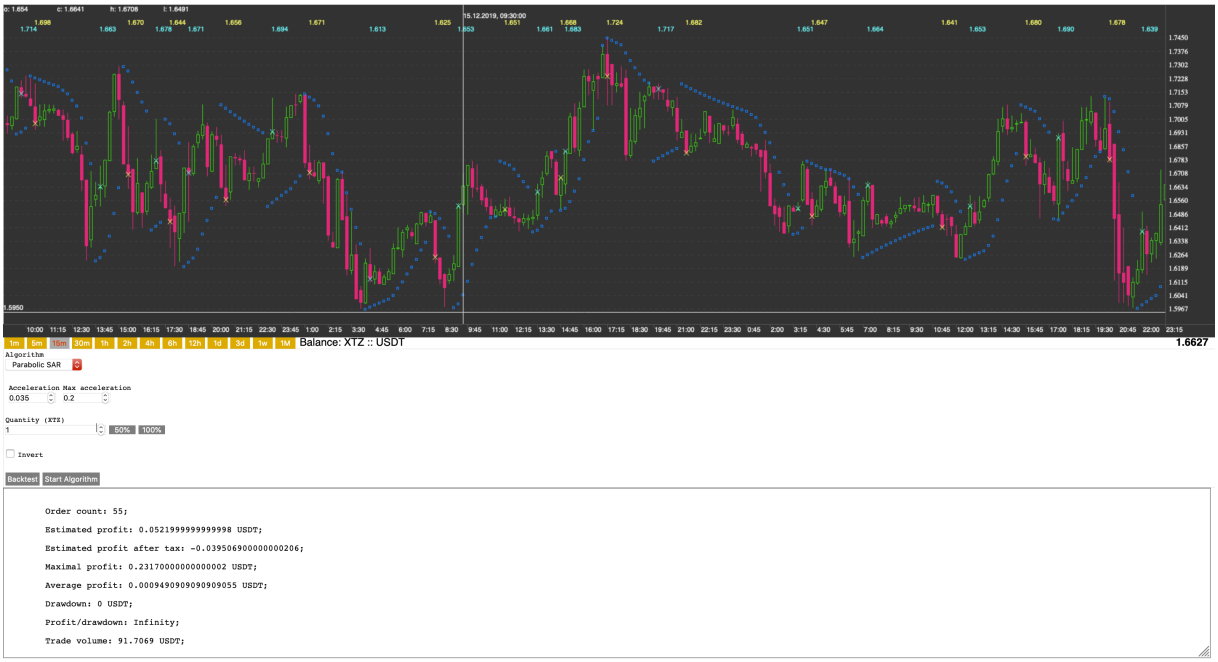
Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА MACD НА П’ЯТНАДЦЯТИХВИЛИННОМУ ІНТЕРВАЛІ



Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

РОБОТА ТА РЕЗУЛЬТАТ ІНДИКАТОРА PSAR П’ЯТНАДЦЯТИХВИЛИННОМУ ІНТЕРВАЛІ



Розроб.	*Карчевський К.Д..*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*	
Перевір.	*Ситніков. Д.Е.*				
Н. Контр.	*Ситніков. Д.Е.*				
				ІТПм-18-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 1*

ДОДАТОК В
Специфікація

№	Обозначення	Найменування	Дод. відо-мості
		Текстові документи	
1	ГЮИК.506440.002 ПЗ	Пояснювальна записка	52 листи
2	ГЮИК.506440.002 – 01 12 01	Текст програми	13 листів

Изм.	Лист	№ докум.	Підпис	Дата	ГЮИК.506440.002			
Розробив		Карчевський К.Д.			Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів		Лист	Листів
Перевірив		Ситніков Д.Е.					1	1
Н. Контр.		Ситніков Д.Е.				ХНУРЭ Кафедра СТ		
Затвердив		Гребеннік І.В.						

ДОДАТОК Г

Відомість дипломної роботи

№	Обозначення	Найменування	Дод. відо-мості
		Текстові документи	
1	ГЮИК.506440.002 ПЗ	Пояснювальна записка	51 лист
2	ГЮИК.506440.002 – 01 12 01	Текст програми	12 листів
		Графічні документи	
3		Робота та результат розробленого індикатора на годинному інтервалі	1 лист
4		Робота та результат індикатора MACD на годинному інтервалі	1 лист
5		Робота та результат індикатора PSAR на годинному інтервалі	1 лист
6		Робота та результат розробленого індикатора на шестигодинному інтервалі	1 лист
7		Робота та результат індикатора MACD на шестигодинному інтервалі	1 лист
8		Робота та результат індикатора PSAR на шестигодинному інтервалі	1 лист
9		Робота та результат розробленого індикатора п’ятнадцятихвилинному інтервалі	1 лист
10		Робота та результат індикатора MACD п’ятнадцятихвилинному інтервалі	1 лист
11		Робота та результат індикатора PSAR на п’ятнадцятихвилинному інтервалі	1 лист

					ГЮИК.506440.002 Д3			
Изм.	*Лист*	*№ докум.*	*Підпис*	*Дата*				
Розробив		*Карчевський К.Д.*			*Розробка методу інтелектуального аналізу даних ринків криптовалют за допомогою комбінації технічних індикаторів*		Лист	Листів
Перевірив		*Ситніков Д.Е.*					*1*	*2*
Н. Контр.		*Ситніков Д.Е.*				*ХНУРЭ Кафедра СТ*		
Затвердив		*Гребеннік І.В.*						