УДК 62-5, 681.5.015, 004.942

Алгоритмы интеллектуализации и их применение в АСУ

А. А. Личман, О. Ю. Чередникова
E-mail: anton.lichman@yandex.ru

Аннотация:
Предложен алгоритм поиска наилучших решений при совершенствовании работы программной части АСУ (автоматизированных систем управления), позволяющий оптимизировать затраты ресурсов АСУ на выполнение различных задач. Анализируются проблемы оптимизации алгоритмов под конкретную задачу. Алгоритм поиска наилучших решений учитывает также затраты времени, и имеет конкретные замеренные результаты, что позволяет делать промежуточные выводы о его эффективности. Приведены результаты исследований.

Введение

С помощью мощностей нейронных сетей и прочих современных развивающихся инструментов анализа можно не только создавать новое, но и совершенствовать уже рабочее старое, что позволяет внедрить интенсивные методы развития в экстенсивные, по большей части консервативные, предприятия. Также современные и довольно значимые предприятия со стандартными АСУ способны улучшить свою работу за счет усовершенствования программной части, что является актуальной в настоящее время задачей. В этих целях будут использоваться алгоритмы интеллектуализации программной части АСУ написанной на C#.

Целью данной статьи является рассмотрение, внедрение и анализ результатов работы алгоритмов интеллектуализации, а именно, внедрение односложных и комбинированных алгоритмов с приоритетом в скорость, или точность вычислений.

Проблемы АСУ и способы их решения

Развитие автоматизированных систем управления представляет собой экспоненциальное улучшение трех уровней (рис.1), совместно или по отдельности.

Рисунок 1 - Трехуровневая система АСУ

Нижним уровнем принято называть датчики и механизмы, систему ввода-вывода информации. Средним уровнем называют микроконтроллеры, отвечающие за запуск и регулирование. Верхним же уровнем является непосредственно программная часть.

В современном мире АСУ постепенно замещается более новой технологией КАП (Комплексное автоматизированное производство) или по английский CIM. В истоках данной технологии прямое продолжение АСУ, с той разницей, что АСУ является системой управления, в то же время как КАП полностью автоматизирует производство.

При должных ресурсах многие сейчас переходят на использование КАП и отдают предпочтение полной автоматизации процесса, однако в ближайшее время эта замена может быть затруднена [1].

Поскольку сейчас на рынке решения, связанные с полной автоматизацией задач неоправданно дорогие из-за санкционной политики в отношении России, идет отказ от неоправданно дорогого повсеместного перехода на КАП, то есть полной автоматизации. Вместо этого производства и кампании стараются искать дешевую альтернативу, ничуть не уступающую КАП в эффективности, а поскольку темпы улучшения датчиков и микроконтроллеров снизились, а цены на них неоправданно дорогие, приходится уповать только на улучшение программной части при помощи алгоритмов интеллектуализации (рис.2).

Развитие программной части Автоматизированных систем управления является одним из ключевых направлений современной технологической индустрии.

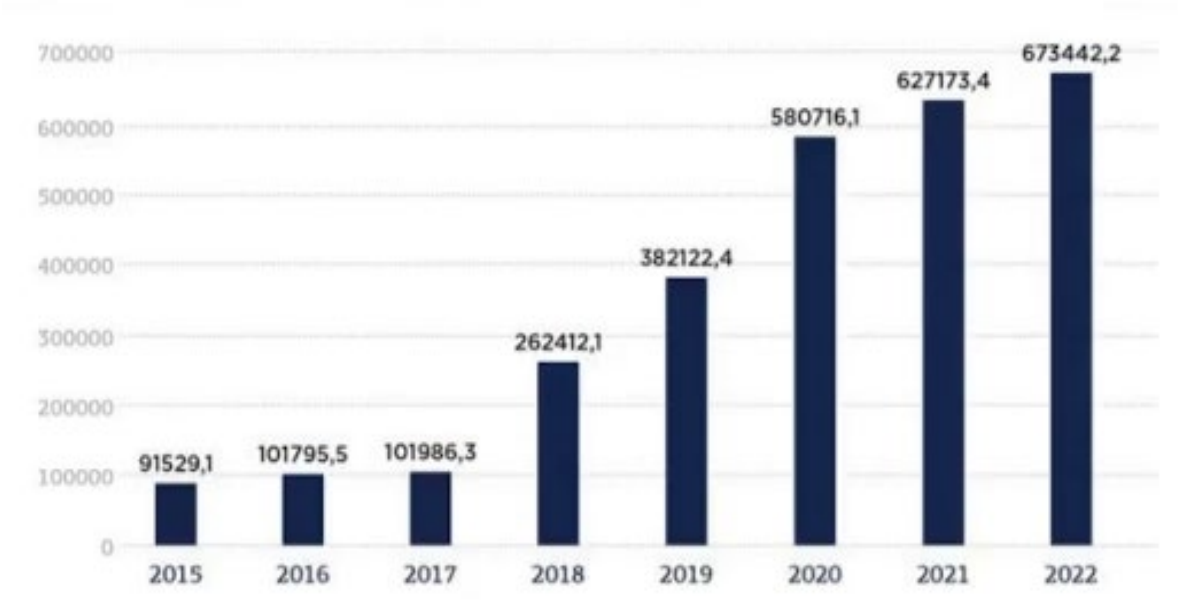

Рисунок 2 -Объем и темпы роста АСУ в России в стоимостном выражении млрд. рублей

Программное обеспечение АСУ представляет собой комплекс специализированных приложений, позволяющих автоматизировать управленческие процессы и повысить эффективность работы предприятий и организаций.

Одним из основных трендов в развитии программной части АСУ является увеличение функциональности и возможностей системы, то есть добавление новых функций к уже существующим. С появлением новых технологий, таких как искусственный интеллект, алгоритмы интеллектуализации и облачные вычисления, и интернет вещей, АСУ получают новые возможности для оптимизации процессов управления и повышения производительности. Внедрение машинного обучения и алгоритмов анализа данных позволяет системам АСУ автоматически принимать решения на основе больших объемов информации.

Кроме того, развитие программной части АСУ связано с улучшением пользовательского опыта. Разработка интуитивно понятных и удобных интерфейсов позволяет пользователям легко взаимодействовать с системой и выполнять необходимые задачи. Постоянное обновление и совершенствование программного обеспечения позволяет АСУ быть актуальными и соответствующими современным требованиям бизнеса.

Таким образом, развитие программной части АСУ направлено на повышение производительности, эффективности и скорости управленческих процессов. Постоянное инновационное развитие в данной области является необходимым условием для успешной автоматизации управления и обеспечения конкурентоспособности организаций.

Многозадачность верхнего уровня АСУ

В АСУ для реализации верхнего уровня повсеместно применяются универсальные и многозадачные программные решения [2]. Это позволяет экономить время и ресурсы, однако

такой подход имеет ряд недостатков. Многозадачные универсальные алгоритмы могут столкнуться с рядом проблем при решении задач АСУ, в которых зачастую многие задачи выполняются одновременно. Вот некоторые из них:

1) Недостаточность ресурсов: Исполнение нескольких задач одновременно требует больших вычислительных ресурсов. Если программа использует большое количество оперативной памяти или процессорного времени, это может привести к замедлению работы компьютера и даже к его зависанию.

2) Проблемы синхронизации: при работе с несколькими задачами одновременно возникает проблема координации между ними. Если не уделить должного внимания синхронизации потоков выполнения, то это может привести к непредсказуемым результатам или даже к ошибкам в работе программы.

3) Проблемы с производительностью: Некоторые многозадачные алгоритмы могут работать менее эффективно, чем их однозадачные аналоги из-за дополнительных затрат на управление многозадачностью.

4) Распределение ресурсов: при работе с несколькими задачами одновременно важно правильно распределить ресурсы между ними. Если какая-то из задач потребляет слишком много ресурсов, это может отрицательно сказаться на выполнении других задач.

В целом многозадачные универсальные программы – это хороший инструмент, несмотря на все недостатки, однако их можно улучшить, минимизировав проблемы при помощи алгоритмов интеллектуализации – совокупности алгоритмов интеллектуального анализа данных и своеобразного узкопрофильного отладчика для исправления найденных ошибок, что дает пространство для улучшения.

Факторы, влияющие на производительность программной части АСУ

Качество АСУ описывается формулой (1).

$$J = \int_{t0}^{tk} G(X, U, F)dt, \quad (1)$$

где [t0, tk] — рассматриваемый интервал времени;

G(...) — функция, отражающая показатель качества;

X, U, F — векторы координат состояний, управлений и возмущений, соответственно.

Показатель качества G вычисляется в зависимости от типа АСУ, но в любом случае Точность и Быстродействие влияют на это значение. Точность (погрешность) измеряется по формуле (2):

$$\delta = \Delta / X\_и, \quad (2)$$

где Δ - абсолютная погрешность измерений;

X_i - истинное значение величины.

В случае, если точность не соответствует нормам, ее положено пересчитывать, изменяя значения коэффициентов формулы.

Не беря в расчет погрешность приборов, проведем эксперимент и замеряем 50 величин в исследуемой АСУ. Замеры были произведены на основе экспериментальной АСУ в сфере гидрогеологии.

Решалась задача установления плотности грунтов, а также наличия грунтовых вод. Величины, измерялись в миллиметрах водного столба.

Плотность грунтовых вод определяется по формуле (3):

$$P = p \cdot g \cdot h, \quad (3)$$

где p - давление слоя жидкости;

g – коэффициент;

h- высота слоя.

P и p - измеряемые величины, на которые нельзя повлиять программными средствами никак, кроме как констатацией их несоответствия, что обычно и так проверяется до ввода данных в программу. Остается лишь коэффициент g, который вычисляется по различным формулам с отличающейся точностью в зависимости от требований задачи.

Оценка производительности АСУ может вычисляться по формуле (4):

$$T = \sum_{i=1}^{n} m_i t_i, \quad (4)$$

где m - число команд;

t – время выполнения;

n - число замеров.

Однако, учитывая временные потери во время пересчета погрешности, быстродействие работы комплекса может уменьшиться [3].

Бенчмарки, как способ измерения производительности приложения

Оценка производительности программной части АСУ будет произведена при помощи бенчмарков из библиотеки BenchmarkDotNet Библиотека доступна как opensource проект на github и поддерживает широкий набор сред выполнения: .NET 5+, .NET Framework 4.6.1+, .NET Core 2.0+, Mono, NativeAOT. Добавляется через NuGET.

Для тестирования все методы должны иметь модификатор public, и к этим методам должен применяться атрибут [Benchmark]. Кроме того, сам класс, который содержит эти методы, должен иметь модификатор public. Потому

программа подсчета была модифицирована под данные условия.

Для запуска теста вызывается метод BenchmarkRunner.Run, который типизируется тестируемым классом:

BenchmarkRunner.Run<StringTest> ();

Для достоверности теста убираем все процессы кроме стандартных windows и запускаем программу не через Visual Studio, а через dotnet cli.

BenchmarkDotNet имеет очень много различных настроек и возможностей конфигурации, с помощью которых будет произведена первичная диагностика.

Применение пермутационной интеллектуализации

С нижних уровней программа получает данные: 50 значений, на основе которых производятся дальнейшие вычисления, и получаются 50 результатов со статистикой. Программа затрачивает определенное количество времени на их вычисление. В ходе применения пермутационной интеллектуализации мы выявим полезные пермутации и внедрим их, комбинируя показатели точности и быстродействия [4].

Пермутация – изменение заданного параметра путем корректировки не константных величин, от которых зависит данный параметр.

Программу запускаем через консоль и определяем бенчмарки: 1- точность вычисления 2- время работы.

BenchmarkDotNet позволяет замерять время, необходимое на весь процесс вычисления, а также результат каждого вычисления в зависимости от значения коэффициента. С помощью диагностики получаем 50 результатов прогонки (рис.3).

Погрешность не может выходить за рамки 0.05 мм.в.ст. Как видно из рис.3 для 13 и 22 измерения требуется пересчет. Пересчет производится вручную путем изменения коэффициента g.

В качестве второго эксперимента запускаем комбинированную оценку алгоритма и пытаемся его усовершенствовать: идет перебор всех возможных пермутаций (изменений) через бенчмарки (критерии), с ограничением по времени, иначе процесс может быть слишком длительным. Приоритет отдадим скорости. Проведем пермутации через бенчмарк по времени работы (рис.4.).

Время работы сократилось значительно, в среднем почти на 10 секунд за счет сокращения ненужных операций, но значительно увеличилась погрешность вычислений - недопустимых значений стало 5. Как ни парадоксально пересчет данных повлек за собой значительные временные штрафы, полностью перекрывающие 10 секунд сэкономленного времени.

№	Погрешность (мм. в. ст.)	Время сек
1	0.03	62
2	0.01	57
3	0.03	59
4	0.02	67
5	0.02	64
6	0.03	61
7	0.04	45
8	0.02	58
9	0.03	54
10	0.01	65
11	0.01	64
12	0.04	66
13	0.08	65

22	0.14	75
23	0.02	62
24	0.04	64

43	0.04	58
44	0.01	64
45	0.05	65
46	0.05	69
47	0.02	61
48	0.01	58
49	0.01	59
50	0.04	65

Рисунок 3 - Результаты первой прогонки

Приходится идти на уступки и принять как факт, что достоверно невозможно узнать идеальный вариант, однако можно найти оптимальный, не перебирая их все [5].

Остановимся на локальном поиске, где сначала определяется метрика, на основе бенчмарков сравнивается два решения и отбрасывается худшее, затем лучшее сравнивается с третьим (соседним решением) и так далее, по окончанию процесса просто выводится лучший вариант.

Данная техника оптимизации программной части АСУТП является эффективной, так как несколько строк кода могут улучшить отдельные показатели в несколько раз. В данном случае можно пренебречь так называемой концепцией «локального оптимума», так как результаты говорят сами за себя, или ждать пока появятся мощности позволяющие обработать все варианты, или все-таки верить в практический подход и улучшение (пусть гипотетически и не самое лучшее) АСУТП.

№	Погрешность (мм. в. ст.)	Время сек
1	0.05	55
2	0.04	58
3	0.04	62
4	0.13	54

13	0.08	67

22	0.14	62

43	0.28	71
44	0.04	46
45	0.05	51
46	0.06	55
47	0.05	66
48	0.03	59
49	0.02	64
50	0.04	52

Рисунок 4 - Результаты второй прогонки

Проверим предложенный вариант пермутаций с приоритетом в точность (рис.5).

№	Погрешность (мм. в. ст.)	Время сек
1	0.02	62

46	0.19	53
47	0.02	62
48	0.01	67
49	0.01	52
50	0.01	64

Рисунок 5 - Результаты третьей прогонки

Как видно из результатов, время работы немного сократилось - примерно на 2 секунды, однако точность значительно выросла. И теперь имеем всего лишь одну погрешность в 46 строке.

В процессе дальнейших прогонок результатов различных экспериментов удалось выяснить, что погрешностей действительно стало меньше, а время работы сократилось [6].

Выводы

Учитывая полученные в ходе эксперимента данные можно сделать вывод, что можно работать с АСУ при помощи алгоритмов интеллектуализации, перебирая необходимые бенчмарки в определенном порядке, что в конечном итоге приведет к появлению «хороших» пермутаций и, как следствие, улучшенной работе программы.

Перед тем, как осуществлять проект внедрения нужно максимально формализовать его цели.

Недостатками подобного подхода является низкая актуальность подобного рода локальных улучшений без высокой плотности работы, в случае достаточно высокой степени автоматизации системы.

Такой подход помогает исключить такие затратные по времени действия как ручной пересчет погрешностей.

Однако же чем больше автоматизирована АСУ, тем менее эффективен данный метод улучшения ее работы.

Следует отметить, что использовать подобные алгоритмы можно даже в комплексном автоматизированном производстве и АСУ с высоким уровнем автоматизации, но это следует делать только при крайней необходимости максимальной оптимизации [7-10].

Литература

1. Трапезников В. А. Автоматизация проектирования систем управления // Трапезников В. А -М.: Финансы и статистика, 2017. - 208 с.

2. Фельдбаум, А. А. Вычислительные устройства в автоматических системах // А.А. Фельдбаум. - М.: Государственное издательство физико-математической литературы, 2017. - 800 с.

3. Сигорский, В. П. Многоустойчивые элементы дискретной техники // В.П. Сигорский, Л.С. Ситников, Л.Л. Утяков. - М.: Энергия, 2017. - 358 с.

4. Воронов, А. Элементы теории автоматического регулирования // А. Воронов. - М.: Воениздат, 2015. - 472 с.

5. Xianjun Ni Research of Data Mining Based on Neural Networks // World Academy of Science, Engineering and Technology. - 2008. - № 39. - P. 381-384.

6. Манжула, В.Г. Методы «мягких» вычислений для аналитической обработки информации в условиях неопределенности / В.Г. Манжула, С.А. Морозов, С.В. Федосеев // Фундаментальные исследования. - 2009. - № 4. – С. 75-76.

7. Назаров А.В., Лоскутов А.И. Нейросетевые алгоритмы прогнозирования и оптимизации систем // А.В.Назаров, А.И. Лоскутов - СПб.: Наука и Техника, 2003. - 384 с.

8. Matthew Mc-Donald WPF: Windows Presentation Foundation в .NET 4.0 with examples C# 2010 for professionals = Pro WPF in C# 2010: Windows Presentation Foundation with .NET 4.0. // Mc-Donald Matthew – М.: « Вильямс», 2011. – 1024 с.

9. Фельдбаум, А. А. Электрические системы автоматического регулирования // А.А. Фельдбаум. - М.: Государственное издательство оборонной промышленности, 2017. - 808 с.

10. Натан. А. WPF 4. Подробное руководство // А. Натан – СПб. : Символ-Плюс, 2011. – 880 с.

Личман А.А. Чередникова О.Ю. Алгоритмы интеллектуализации и их применение в АСУ. *Предложен алгоритм поиска наилучших решений при совершенствовании работы программной части АСУ (автоматизированных систем управления), позволяющий оптимизировать затраты ресурсов АСУ на выполнение различных задач. Анализируются проблемы оптимизации алгоритмов под конкретную задачу. Алгоритм поиска наилучших решений учитывает также затраты времени, и имеет конкретные замеренные результаты, что позволяет делать промежуточные выводы о его эффективности. Приведены результаты исследований.*

Ключевые слова: *АСУ, бэнчмарки, пермутации, эффективность.*

Lichman A.A. Cherednikova O.Yu. Intellectualization algorithms and their application in automated control systems*. An algorithm is proposed for finding the best solutions when improving the operation of the software part of ACS (automated control systems), which allows optimizing the expenditure of ACS resources to perform various tasks. The problems of optimizing algorithms for a specific task are analyzed. The algorithm for finding the best solutions also takes into account the time spent and has specific measured results, which makes it possible to draw intermediate conclusions about its effectiveness. The results of research are done.*

Keywords: *ACS, benchmarks, permutations, efficiency.*

Статья поступила в редакцию 10.06.2024
Рекомендована к публикации профессором Мальчевой Р. В.