УДК 004.51

В. А. Иванова – студентка кафедры информационных систем

ПОСТРОЕНИЕ СВЯЗЕЙ МЕЖДУ ОБЪЕКТАМИ ДЛЯ ПО СОЗДАНИЯ ВЫСОКОУРОВНЕВОГО АЛГОРИТМИЧЕСКОГО ОПИСАНИЯ СИСТЕМЫ ПАРАЛЛЕЛЬНЫХ ВЗАИМОДЕЙСТВУЮЩИХ ПРОЦЕССОВ

Программное обеспечение для создания высокоуровневого алгоритмического описания системы параллельных взаимодействующих процессов [1] в широком смысле является редактором диаграмм. В ходе его разработки мы столкнулись с задачей, которая встает перед разработчиком любого программного обеспечения, позволяющего создавать и редактировать блок-схемы. Эта задача – построение связей между объектами, размещенными пользователем на рабочем поле.

Алгоритм построения связей между объектами диаграммы должен обладать следующими основными свойствами:

— обрабатывать большое поле поиска. Под полем поиска понимается рабочее поле, на которое пользователь выносит объекты. Алгоритм должен поддерживать построение связей на больших полях;

— обеспечивать высокую скорость. Связь должна рассчитываться и отрисовываться с малой задержкой (субъективно – не превышающей ¼ секунды, то есть время, когда перестраивающийся путь не будет мерцать на экране).

Сразу можно сказать, что переборные алгоритмы с экспоненциальной сложностью не удовлетворяют указанным выше требованиям. В ходе исследования было установлено, что подобному алгоритму требуется существенное время (порядка несколько десятков секунд), чтобы найти путь между объектами на поле размером 15x15 клеток, что является неприемлемым. Следовательно, необходимо использовать эвристические алгоритмы, имеющие линейную сложность.

Основной проблемой эвристических алгоритмов является не гарантированное и, в большинстве случаев, не поддающееся аналитической оценке качество результатов. Наиболее часто используемыми для построения маршрутов (а задача построения связи, фактически, и является задачей построения маршрута) алгоритмами являются жадный и волновой алгоритмы. Эти алгоритмы являются алгоритмами с линейной сложностью и, соответственно, позволяют построить маршрут за требуемое время. Эти два алгоритма были выбраны в качестве кандидатов на роль базовых для разрабатываемого алгоритма.

Как было сказано ранее, нами был выбран линейный алгоритм. Встал выбор между двумя алгоритмами нахождения путей – жадным и волновым. При заманчивой простоте реализации, жадный алгоритм дает слишком непредсказуемые результаты. Путь, найденный таким алгоритмом, не всегда будет минимальным. В отличие от жадного, волновой алгоритм всегда находит минимальный путь [2]. Таким образом, волновой алгоритм, вроде бы, является решением поставленной задачи.

После исследования большого количества результатов построения связей, получаемых с помощью волнового алгоритма, был сделан вывод о том, что далеко не всегда применение формального критерия минимальной длины связи дает лучший с точки зрения восприятия пользователем результат. Результат работы алгоритма, отрисованный на экране, не должен вызывать у пользователя желания что-то подправить самостоятельно. Т.е. при минимальных усилиях должны получаться хорошие, естественные для визуального восприятия результаты. В частности, для человека в ряде случаев, более предпочтительным является путь с меньшим количеством изгибов. В ряде случаев предпочтительным является пересечение других объектов диаграммы вместо их обхода и т.д.

В итоге было сформулировано, что кроме минимизации длины связи алгоритм должен учитывать следующие визуальные факторы:

— алгоритм должен строить пути, как с обходом препятствий, так и с их пересечением;

— алгоритм должен по возможности минимизировать количество поворотов;

— алгоритм должен минимизировать наложение связи на другие связи и пересечения с другими связями.

Поскольку, мы остановились на волновом алгоритме в качестве базового, то для того, чтобы он удовлетворял заявленным выше требованиям, потребовалась его существенная модификация.

Итак, первая задача – обход препятствий. Волновой алгоритм по умолчанию обходит любое препятствие, отмеченное на карте. Рассмотрим для сравнения два варианта построения пути между препятствием.

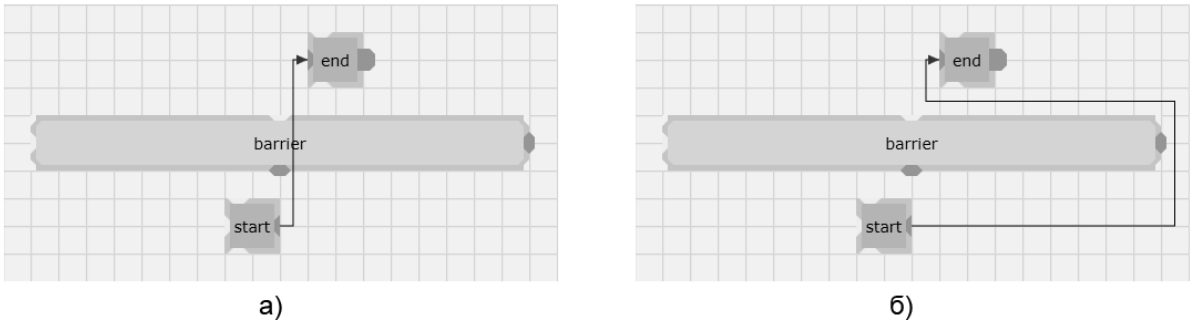

Рис. 1. Варианты обхода и пересечения препятствия

Очевидно, что вариант, показанный на рис. 1,а – путь, пересекающий большое препятствие, воспринимается существенно лучше, чем путь, огибающий его (рис. 1,б). Таким образом, нам предстоит решить проблему определения необходимости пересечения препятствия.

Вторая задача – предотвращение наложения связей друг на друга (рис. 2).При наложении нескольких связей друг на друга затрудняется их визуальное восприятие и увеличивается время на редактирование схемы (рис. 2,а). Если в качестве решения принимать все ранее отрисованные связи за препятствия, то на рабочем поле в скором времени закончится свободное место.

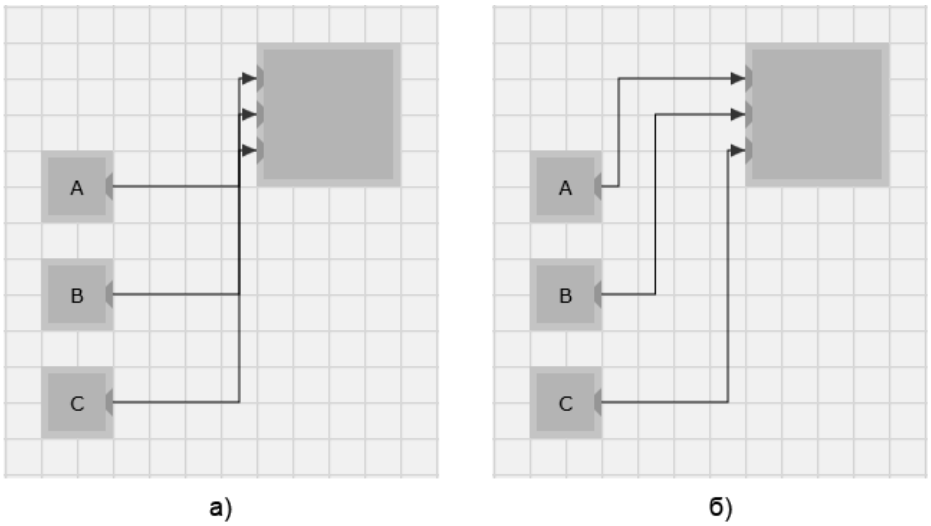

Рис. 2. Варианты с наложением связей и без наложения

Последняя задача – выбор пути с минимальным количеством поворотов. На иллюстрации (рис. 3) представлены два варианта отрисовки пути. Путь с меньшим количеством поворотов (рис. 3, б) оказывается не минимальным, хоть и очевидно, что он визуально предпочтительней, чем путь с большим количеством поворотов (рис. 3, а).

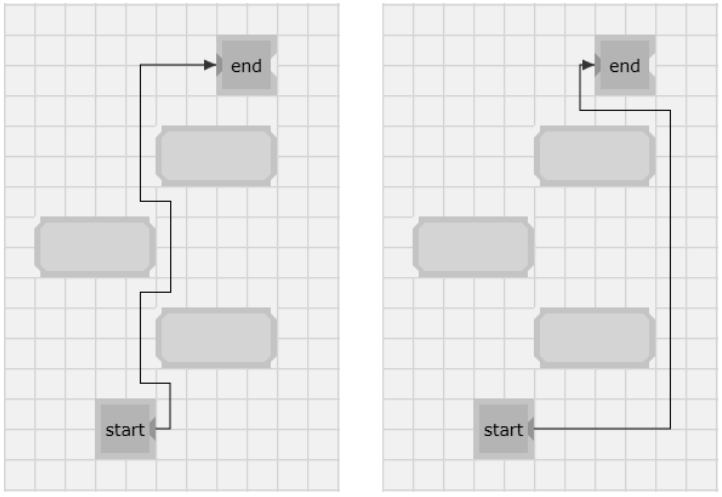

Рис. 3. Варианты с большим и минимальным количеством поворотов

Продемонстрируем суть модификации волнового алгоритма на примере аналогии резервуара с водой (что и дало название исходному алгоритму). Поведение классического волнового алгоритма соответствует физическим процессам лишь частично. Фактически, разработанный модифицированный волновой алгоритм просто учитывает большее количество физических процессов. Поместим наши объекты блок-схемы в условный резервуар, в который будет волнами поступать вода (рис. 4,а). Искомый маршрут это тот путь, по которому волна достигнет требуемой точки из исходной. Представим, что наш объект бесконечно высок (модель классического волнового алгоритма). Тогда, каким бы не был уровень воды в резервуаре, волна всегда будет достигать финишной точки, огибая объект. Понизим высоту объекта с бесконечности до некоторого конкретного числа. При определенной высоте объекта волна достигнет финишной точки быстрее, перехлестнув через него, чем огибая (рис. 4,б).

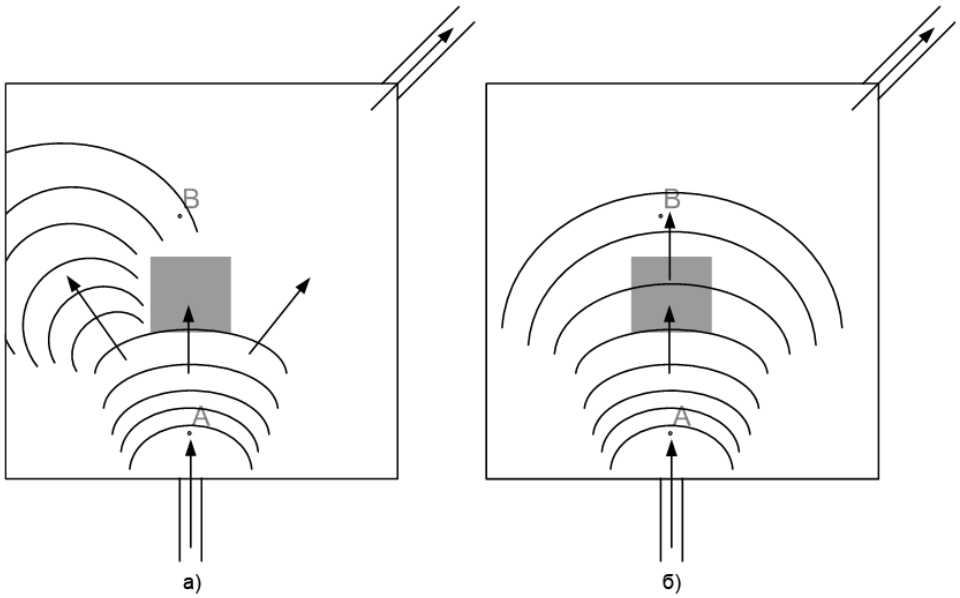

Рис. 4. Схема распространения волн

Введем в волновой алгоритм коэффициенты, характеризующие уровни объекта. Регулируя эти коэффициенты, мы сможем добиться хороших визуальных результатов. При этом для различных классов объектов диаграмм можно назначать различные коэффициенты высоты, более тонко настраивая поведение алгоритма построения связей.

Проблема наложения связей имеет аналогичное решение. Если изначально уровень связей соответствовал уровню дна резервуара, теперь следует назначить им некий больший уровень. При этом

следует отметить, что пересечение связей (в отличие от пересечения объектов) это приемлемый вариант для большинства схем. Следовательно, коэффициент высоты связей должен быть значительно меньше, чем объектов диаграмм.

Для решения задачи с поворотами вводим понятие «направление» для клетки и дополнительный коэффициент – своего рода «стоимость поворота». Если волна распространяется в направлении, соответствующем направлению очередной клетки на карте, коэффициент не прибавляется, в обратном случае – прибавляется. Таким образом, извилистый путь становится «дороже» прямого.

Процесс распространения волны от клетки-источника (**source_cell**) выглядит следующим образом. Волна распространяется в 4 направлениях: на клетку вверх, на клетку вниз, на клетку влево и на клетку вправо. Обозначим очередное направление как **direction**, а значение, которое мы будем заносить в очередную соседнюю клетку **cell**, обозначим как **value**.

cell.value = source_cell.value + cell.coefficient + rotation,

где **cell.coefficient** – уровень клетки над рабочим полем, **rotation** – значение поворота.

Значение поворота определяется следующим образом:

```
if (source_cell.Direction != direction)
rotation = RotationCoef;
else
rotation = 0;
```

где **RotationCoef** – коэффициент «стоимость поворота».

Если направление клетки **source_cell** соответствует направлению, в котором мы двигаемся, проставляя новые значения, то значение коэффициента будет равно нулю. Если же направления различаются – значение коэффициента будет ненулевым (рис. 5).

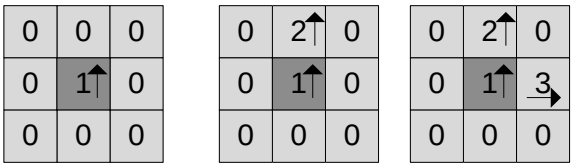

Рис. 5. Вычисление значения клеток карты

Если получившееся значение больше того, что ранее стояло в клетке cell, заменяем старое значение новым.

После того, как волна достигнет точки назначения, начинается непосредственно построение пути по заполненной волной территории. При построении пути на размеченной карте мы двигаемся по карте от финишной точки к стартовой. Проверяем клетки в четырех направлениях от финишной. Выбираем клетку с наименьшим ненулевым значением и движемся в выбранном направлении.

Как выбирается любая следующая клетка пути:

```
if (cell.Value == source_cell.Value - source_cell.Coefficient - source_cell.rotation)
{
cell.AddToPath();
}
```

Зная клетки карты, в которых находится путь, нетрудно отрисовать его на рабочем поле.

После всех модификаций сложность волнового алгоритма осталась линейной, что подтверждает правильность нашего выбора. Полученные результаты практически не требуют от пользователя вмешательства и корректировок – т.е. удовлетворяют визуальным требованиям, поставленным нами перед алгоритмом. Техника использования настраиваемых коэффициентов позволяет гибко настраивать действие алгоритма под требования конкретной задачи.

Библиографический список

1. А.Ю. Сыщиков, Б.Н. Седов, Концепция дизайна интегрированной среды разработки для визуального программирования. Научная сессия ГУАП: Сб. докл.: В 3 ч. Ч. I. Технические науки/ ГУАП. СПб., 2012,
2. Арсеньев А. А. Обход препятствий: волновой алгоритм. http://www.ironfelix.ru/modules.php?name=Pages&pa=showpage&pid=96