Find the best combination from a given set of multiple sets

Asked 16 years, 4 months ago Modified 12 years, 5 months ago Viewed 2k times

Part of PHP Collective

10

Say you have a shipment. It needs to go from point A to point B, point B to point C and finally point C to point D. You need it to get there in five days for the least amount of money possible. There are three possible shippers for each leg, each with their own different time and cost for each leg:

```
Array
(
    [leg0] => Array
        (
            [UPS] => Array
                (
                    [days] => 1
                    [cost] => 5000
                )

            [FedEx] => Array
                (
                    [days] => 2
                    [cost] => 3000
                )

            [Conway] => Array
                (
                    [days] => 5
                    [cost] => 1000
                )

        )

    [leg1] => Array
        (
            [UPS] => Array
                (
                    [days] => 1
                    [cost] => 3000
                )

            [FedEx] => Array
                (
                    [days] => 2
                    [cost] => 3000
                )

            [Conway] => Array
                (
                    [days] => 3
                    [cost] => 1000
```

```
                  )

          )

    [leg2] => Array
        (
            [UPS] => Array
                (
                    [days] => 1
                    [cost] => 4000
                )

            [FedEx] => Array
                (
                    [days] => 1
                    [cost] => 3000
                )

            [Conway] => Array
                (
                    [days] => 2
                    [cost] => 5000
                )

        )

)
```

How would you go about finding the best combination programmatically?

My best attempt so far (third or fourth algorithm) is:

1. Find the longest shipper for each leg
2. Eliminate the most "expensive" one
3. Find the cheapest shipper for each leg
4. Calculate the total cost & days
5. If days are acceptable, finish, else, goto 1

Quickly mocked-up in PHP (note that the test array below works swimmingly, but if you try it with the test array from above, it does not find the correct combination):

```
$shippers["leg1"] = array(
    "UPS"    => array("days" => 1, "cost" => 4000),
    "Conway" => array("days" => 3, "cost" => 3200),
    "FedEx"  => array("days" => 8, "cost" => 1000)
);

$shippers["leg2"] = array(
    "UPS"    => array("days" => 1, "cost" => 3500),
    "Conway" => array("days" => 2, "cost" => 2800),
    "FedEx"  => array("days" => 4, "cost" => 900)
);

$shippers["leg3"] = array(
    "UPS"    => array("days" => 1, "cost" => 3500),
    "Conway" => array("days" => 2, "cost" => 2800),
    "FedEx"  => array("days" => 4, "cost" => 900)
);

$times = 0;
$totalDays = 9999999;

print "<h1>Shippers to Choose From:</h1><pre>";
print_r($shippers);
print "</pre><br />";

while($totalDays > $maxDays && $times < 500){
            $totalDays = 0;
            $times++;
            $worstShipper = null;
            $longestShippers = null;
            $cheapestShippers = null;

            foreach($shippers as $legName => $leg){
                //find longest shipment for each leg (in terms of days)
                unset($longestShippers[$legName]);
                $longestDays = null;

                if(count($leg) > 1){
                    foreach($leg as $shipperName => $shipper){
                        if(empty($longestDays) || $shipper["days"] >
$longestDays){
                            $longestShippers[$legName]["days"] =
$shipper["days"];
                            $longestShippers[$legName]["cost"] =
$shipper["cost"];
                            $longestShippers[$legName]["name"] = $shipperName;
                            $longestDays = $shipper["days"];
                        }
                    }
                }
            }

            foreach($longestShippers as $leg => $shipper){
                $shipper["totalCost"] = $shipper["days"] * $shipper["cost"];

                //print $shipper["totalCost"] . " &lt;?&gt; " .
$worstShipper["totalCost"] . ";";

                if(empty($worstShipper) || $shipper["totalCost"] >
$worstShipper["totalCost"]){
                    $worstShipper = $shipper;
                    $worstShipperLeg = $leg;
                }
            }

            //print "worst shipper is: shippers[$worstShipperLeg]
[{$worstShipper['name']}]" . $shippers[$worstShipperLeg][$worstShipper["name"]]
["days"];
            unset($shippers[$worstShipperLeg][$worstShipper["name"]]);

            print "<h1>Next:</h1><pre>";
            print_r($shippers);
            print "</pre><br />";

            foreach($shippers as $legName => $leg){
                //find cheapest shipment for each leg (in terms of cost)
                unset($cheapestShippers[$legName]);
                $lowestCost = null;

                foreach($leg as $shipperName => $shipper){
                    if(empty($lowestCost) || $shipper["cost"] < $lowestCost){
                        $cheapestShippers[$legName]["days"] = $shipper["days"];
                        $cheapestShippers[$legName]["cost"] = $shipper["cost"];
                        $cheapestShippers[$legName]["name"] = $shipperName;
                        $lowestCost = $shipper["cost"];
                    }
                }

                //recalculate days and see if we are under max days...
                $totalDays += $cheapestShippers[$legName]['days'];
            }
            //print "<h2>totalDays: $totalDays</h2>";
        }

        print "<h1>Chosen Shippers:</h1><pre>";
        print_r($cheapestShippers);
        print "</pre>";
```

I think I may have to actually do some sort of thing where I literally make each combination one by one (with a series of loops) and add up the total "score" of each, and find the best one....

EDIT: To clarify, this isn't a "homework" assignment (I'm not in school). It is part of my current project at work.

The requirements (as always) have been constantly changing. If I were given the current constraints at the time I began working on this problem, I would be using some variant of the A* algorithm (or Dijkstra's or shortest path or simplex or something). But everything has been morphing and changing, and that brings me to where I'm at right now.

So I guess that means I need to forget about all the crap I've done to this point and just go with what I know I should go with, which is a path finding algorithm.

PHP php algorithm puzzle combinations np-complete

Share

Improve this question

Follow

edited Sep 26, 2008 at 18:27

Hank Gay
71.8k ● 36 ● 161 ● 222

asked Aug 18, 2008 at 16:39

cmcculloh
48.6k ● 42 ● 108 ● 137

+1 for approaching morphing-over-time requirements with a "let's ditch this crap and start over" state of mind. – Alex Jul 3, 2012 at 15:12

7 Answers

Sorted by: Highest score (default)

8

✓

Could alter some of the shortest path algorithms, like Dijkstra's, to weight each path by cost but also keep track of time and stop going along a certain path if the time exceeds your threshold. Should find the cheapest that gets you in under your threshold that way

Share Improve this answer Follow

answered Aug 18, 2008 at 16:52

Kevin Sheffield
3,616 ● 2 ● 25 ● 22

5

Sounds like what you have is called a "linear programming problem". It also sounds like a homework problem, no offense.

The classical solution to a LP problem is called the "Simplex Method". Google it.

However, to use that method, you must have the problem correctly formulated to describe your requirements.

Still, it may be possible to enumerate all possible paths, since you have such a small set. Such a thing won't scale, though.

Share Improve this answer Follow

answered Aug 18, 2008 at 16:48

Baltimark
9,272 ● 12 ● 38 ● 35

5

Sounds like a job for Dijkstra's algorithm:

> Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959, 1 is a graph search algorithm that solves the single-source shortest path problem for a graph with non negative edge path costs, outputting a shortest path tree. This algorithm is often used in routing.

There are also implementation details in the Wikipedia article.

Share Improve this answer Follow

answered Aug 18, 2008 at 16:49

Theo
133k ● 22 ● 167 ● 213

3

If I knew I only had to deal with 5 cities, in a predetermined order, and that there were only 3 routes between adjacent cities, I'd brute force it. No point in being elegant.

If, on the other hand, this were a homework assignment and I were supposed to produce an algorithm that could actually scale, I'd probably take a different approach.

Share Improve this answer Follow

answered Aug 18, 2008 at 16:56
Derek Park
46.8k ● 16 ● 59 ● 76

2

As Baltimark said, this is basically a Linear programming problem. If only the coefficients for the shippers (1 for included, 0 for not included) were not (binary) integers for each leg, this would be more easily solveable. Now you need to find some (binary) integer linear programming (ILP) heuristics as the problem is NP-hard. See Wikipedia on integer linear programming for links; on my linear programming course we used at least Branch and bound.

Actually now that I think of it, this special case is solveable without actual ILP as the amount of days does not matter as long as it is <= 5. Now start by choosing the cheapest carrier for first choice (Conway 5:1000). Next you choose yet again the cheapest, resulting 8 days and 4000 currency units which is too much so we abort that. By trying others too we see that they all results days > 5 so we back to first choice and try the second cheapest (FedEx 2:3000) and then ups in the second and fedex in the last. This gives us total of 4 days and 9000 currency units.

We then could use this cost to prune other searches in the tree that would by some subtree-stage result costs larger that the one we've found already and leave that subtree unsearched from that point on. This only works as long as we can know that searching in the subtree will not produce a better results, as we do here when costs cannot be negative.

Hope this rambling helped a bit :).

Share Improve this answer Follow

answered Sep 19, 2008 at 19:52
Eonwe
31 ● 1 ● 4

2

This is a knapsack problem. The weights are the days in transit, and the profit should be $5000 - cost of leg. Eliminate all negative costs and go from there!

Share
Improve this answer
Follow

edited Jul 3, 2012 at 15:05
user142162

answered Aug 22, 2008 at 22:04
Mat Noguchi
1,080 ● 6 ● 7

-1

I think that Dijkstra's algorithm is for finding a shortest path.

cmcculloh is looking for the minimal cost subject to the constraint that he gets it there in 5 days.

So, merely finding the quickest way won't get him there cheapest, and getting there for the cheapest, won't get it there in the required amount of time.

Share Improve this answer Follow

answered Aug 18, 2008 at 16:56
Baltimark
9,272 ● 12 ● 38 ● 35