# Задача о потоке минимальной стоимости

Стоит на центральном месте среди моделей сетевой оптимизации. Как и задача о кратчайшем пути, рассматривает стоимость (расстояние) для потока через дугу. Может рассматривать несколько источников (поставляющих узлов) и несколько стоков (принимающих узлов) потока, и связанные с ними расходы. Ранее изученные задачи являются частными случаями задачи о потоке минимальной стоимости. Она имеет эффективное решение, так как формулируется в виде задачи линейного программирования, поэтому может быть решена симплекс-

**Условия**

1. Сеть *ориентирована и связана*.
2. По *крайней мере один узел является источником*.
3. По *крайней мере  один узел является  стоком*.
4. Остальные узлы – передающие узлы.
5. Поток дуги допускается в направлении, указанном стрелкой, максимальный поток задается пропускной способностью дуги. Поток в двух направлениях (пары дуг направлены в противоположные стороны).
6. Сеть имеет достаточно дуг с достаточной пропускной способностью, чтобы обеспечить прохождение потока из источника в сток.
7. Стоимость потока через каждую дугу пропорционально количеству потока, где известна стоимость на единицу потока.
8. Цель: минимизация совокупной стоимости прохождения через сеть, при условии удовлетворения спроса.

**Применения:**

Работа распределительных сетей компании (включает в себя определение плана доставки товара от поставщиков (заводов, и т.д.) к промежуточным складам, а затем к клиентам.

**Формулирование модели**

В ориентированной и связанной сети n узлов включая минимум один узел источник и минимум один узел сток. Переменные решения:

***Xij*** *= поток через дугу i □ j,*

Данная информация включает:

***cij*** *= стоимость на единицу потока через i □ j,*

***uij*** *= пропускная способность по дуге i □ j,*

***bi*** *= поток сети из узла* ***i****.*

Значение *bi зависит от природы узла* ***i****, где:*

***bi > 0*** *если узел* ***i*** *источник,*

***bi < 0*** *если узел* ***i*** *сток*

Цель: минимизация совокупной стоимости прохождения через сеть, при условии удовлетворении спроса. Суммирование ведется только по существующим дугам, формулировка задачи линейного программир

Минимизировать $$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

согласно $$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i,$$

*Для каждого узла* ***i,*** и ***0 ≤ xij ≤ uij****, для каждой дуги i □ j.* Первое суммирование в ограничениях узла *представляет общий поток из узла i,* второе суммирование *представляет общий поток в узел i,*

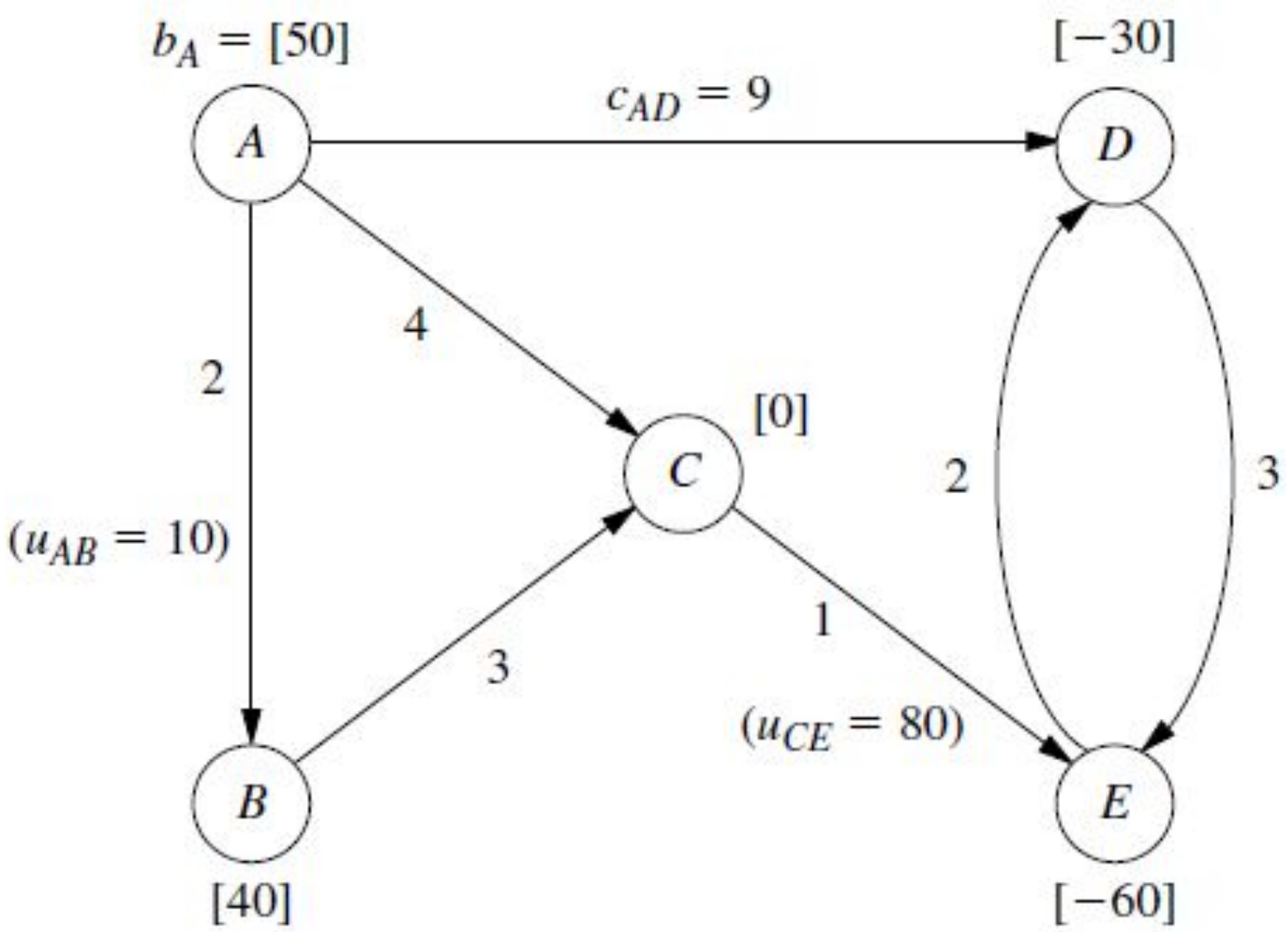
Необходимо иметь нижнюю границу ***Lij > 0*** *для потока по каждой арке* ***i□ j****. Когда это имеет место, используем трансляцию переменных* ***xij'= xij- Lij****, with* ***xij'+ Lij*** *подставленных для* ***xij*** *в модель, чтобы преобразовать модель обратно в упомянутый формат* с ограничениями неотрицательности. Нет гарантии, что задача будет иметь допустимое решение*, частично это зависит от того, какие дуги представлены в сети* и от пропускных способностей дуг. Главное условие: $\sum_{i=1}^{n} b_i = 0.$

**Свойство допустимых решений:** необходимое условие чтобы задача потока с минимальной стоимостью с движением стоимости имела допустимое решение:

Если значения *bi, данные для некоторых приложений* нарушают это условие, обычно говорят, что либо предложение либо спрос представляют собой верхние границы, а не точные величины. Нужно добавить либо фиктивный сток чтобы поглотить избыточное предложение (при ***cij = 0*** к этому узлу добавляются дуги от всех источников) или должен быть добавлен фиктивный источник, чтобы создать поток для избыточного спроса (при ***cij = 0*** от этого узла добавляются дуги ко всем узлам стокам). Для многих приложений, ***bi*** *и* ***uij*** будут иметь целые значения, и реализация задачи потребует, чтобы количество потока ***xij*** было целым числом. Этот результат гарантирован и без введения ограничения целочисленности для переменные благодаря следующему свойству:

**Свойство целочисленного решения** : для задачи минимальной стоимости потока, где каждая ***bi*** и ***uij*** имеют целые значения, все базисные переменные в каждом базисном допустимом решении (в том числе оптимальном)

**Пример:** Сеть распределения. Количества дают значения ***bi, cij,*** и ***uij***. Значения ***bi*** даны в квадратных скобках около узлов, так что узлы источники (поставщики) (***bi > 0)*** , *A и B* (два завода компании), узлы стоки (***bi <0***) - D и E (два склада), и один передающий узел
***(bi = 0)*** - C (распределительный центр). Значения ***cij*** показаны рядом с дугами. Все дуги, за исключением двух имеют пропускные способности, превышающие суммарный поток (90), поэтому ***uij = ∞*** *для всех практических целей*. Две дуги-исключения *A□ B, где*
***uAB = 10****, и дуга C□ E,* для которой ***uCE = 80****.* Модель линейного программирования:
Минимизировать ***Z = 2xAB + 4xAC + 9xAD + 3xBC + xCE + 3xDE + 2xED,***
согласно
***xAB + xAC + xAD = 50***
***-xAB + xBC = 40***
***- xAC - xBC + xCE = 0***




Задача распределения, сформулированная как задача потока минимальной стоимости.

Каждая переменная имеет два ненулевых коэффициента, *1* и *-1*. Эта модель повторяется *в каждой задаче о минимальной стоимости потока*, эта особая структура приводит к целочисленному свойству решений. Любое ограничение узла является *избыточным* (так как сумма всех этих ограничений уравнений дает нули с обеих сторон (предполагая, что допустимые решения существуют, так что сумма значений *bi сводится* к нулю), так что отрицательные уравнения = сумма остальных уравнений. С помощью всего лишь *n – 1* ограничений узла (не избыточных), эти уравнения дают только *n - 1* базисные переменные для ОД решения. Сетевой симплекс-метод рассматривает ***xij ≤ uij*** ограничения как зеркальное отражение *ограничений неотрицательности, поэтому базисные переменные общего числа = n –1* □ прямое соответствие между *n - 1* дугами остового дерева и

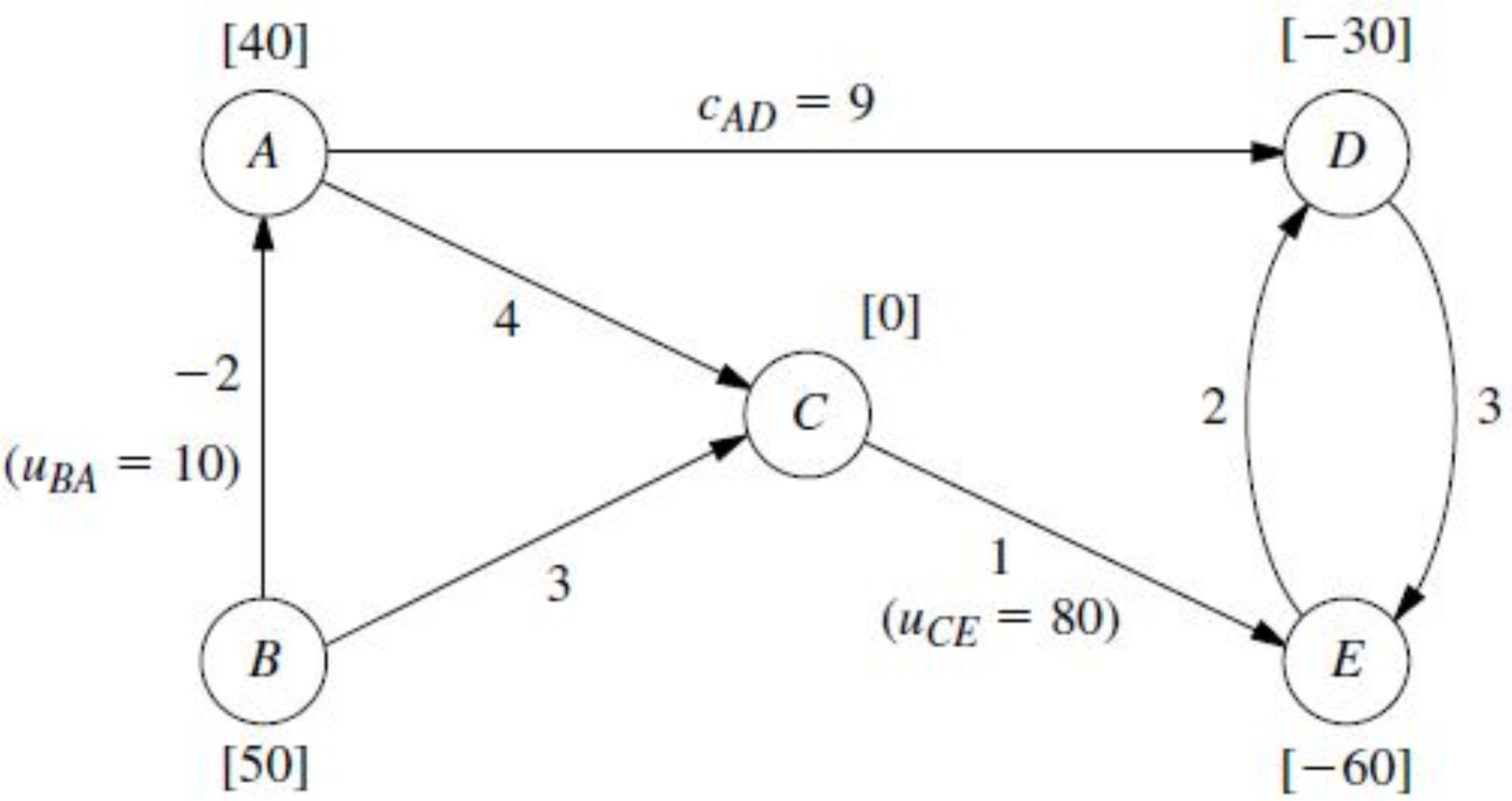
# СЕТЕВОЙ СИМПЛЕКС-МЕТОД

Сетевой симплекс-метод (вариант симплекс-метода) для решения задачи о потоке минимальной стоимости. Он проходит через те же основные шаги при каждой итерации: поиск введенных базисных переменных, определение уходящих базисных переменных, и нахождение нового ОД решения, чтобы перейти от текущего ОД решения к смежному - лучшему (выполняет эти шаги используя специальную структуру сети без необходимости использования симплекс таблицы).

**Использование техники верхней границы:** для эффективного использования ограничения пропускной способности дуг $x_{ij} \le u_{ij}$. Таким образом, вместо того, чтобы рассматривать эти ограничения как функциональные ограничения, они рассматриваются, как ограничения неотрицательности. Поэтому, они рассматриваются т только тогда, когда определена уходящая базисная переменная. Введенная базисная переменная увеличивается от нуля, уходящая базисная переменная - первая базисная переменная, которая достигает либо нижней границы (0) либо верхней границы ($u_{ij}$). Небазисная переменная на ее верхней границе $\boldsymbol{x_{ij} = u_{ij}}$ заменяется на $\boldsymbol{x_{ij} = u_{ij} - y_{ij}}$, так $\boldsymbol{y_{ij} = 0}$ становится небазисной переменной. $y_{ij}$ имеет свою важную интерпретацию в сети. Всякий раз, когда $y_{ij}$ становится базисной переменной со строго положительным значением ($\le u_{ij}$), это значение можно рассматривать как поток от узла $j$ к узлу $i$ (в "неправильном" направлении по дуге $i \to j$), что, в действительности отменяет количество ранее назначенного потока ($x_{ij} = u_{ij}$) от узла $i$ к узлу $j$. Таким образом, когда $x_{ij} \; u_{ij}$ заменяется на $x_{ij} = u_{ij} - y_{ij}$, мы также заменяем реальную дугу $i \to j$ на обратную дугу $j \to i$, где эта новая дуга имеет пропускную способность $u_{ij}$(максимальное количество потока $x_{ij} = u_{ij}$, которое можно отменить) и единичную

Чтобы отобразить поток *xij = uij* через удаленную дугу, мы сдвигаем количество потока, идущее от узла *i* к узлу *j* путем уменьшения *bi* на *uij* и увеличивая *bj на uij*. Если *yij* станет уходящей базисной переменной, достигнув верхней границы, тогда *yij = uij* заменяется на *yij = uij - xij где xij = 0* – новая небазисная переменная, поэтому процесс описанный выше будет обратным (заменяем дуги *j* □ *i* на дугу *i* □ *j* и т.д.) до исходной конфигурации.

Сетевой симплекс метод генерирует последовательность ОД решений, предположим, что *xAB* стала базисной уходящей переменной для некоторой итерации, достигнув верхней границы 10. Следовательно, *xAB* = 10 заменяется на *xAB = 10 - yAB*, и *yAB = 0* становится новой небазисной переменной.




Скорректированная сеть в случае когда метод верхней границы привел к замене *xAB = 10 на xAB = 10 -*

дугу *A* □ B заменяют на дугу *B* □ *A* (с величиной потока *yAB*), для новой дуги назначаем пропускную способность 10 и единичную стоимость -2. Для того, чтобы учесть *xAB = 10*, мы также уменьшаем *bA* с 50 до 40 и увеличиваем *bB* с 40 до 50. Начинаем с *yAB = 0 (xAB = 10)* в качестве небазисной переменной. Следующая итерация покажет, что *xCE* достигла верхней границы 80 и заменяется на *xCE = 80 - yCE*, и так далее, а затем на следующей итерации *yAB* достигает верхней границы 10. Все эти операции выполняются непосредственно в сети, поэтому нам не нужно использовать ярлыки *xij или yij* для потоков дуги или отслеживать, какие дуги являются реальными, а какие - обратными (кроме случаев, когда мы записываем окончательное решение) .
Использование метода верхней границы оставляет

Как правило, задача минимальной стоимости потока, включает  больше дуг, чем узлов  функциональные ограничения это небольшая часть того, что было бы, если бы были включены ограничения пропускных способностей  дуг. Время для вычисления симплекс-методом быстро увеличивается с увеличением числа функциональных ограничений, но с ростом числа переменных (или числа границ ограничений на эти переменные)увеличивается достаточно медленно. Метод верхней границы обеспечивает огромную экономию времени вычислений. Этот метод не используется для задач минимальной стоимости потока, где нет ограничений пропускных способностей дуг.

**Связь между ОД решениями и допустимыми связующими деревьями**

Самое важное понятие сетевого симплекс-метода - сетевое представление ОД решений. С *n* узлами, каждое ОД решение имеет (*n-1*) базисную переменную, где каждая базисная переменная *xij* представляет поток через дугу *i* □ *j*. Эти (*n - 1*) дуги называют **базисными дугами**. (Дуги, соответствующие небазисным переменным *xij = 0 или yij = 0* называют **небазисными дугами**). Базисные дуги никогда не образуют неориентированных циклов (это свойство обеспечивает то, что полученные решения будут являться средним другой пары допустимых решений, что нарушает одно из общих свойств ОД решений).

Любой полный набор *n - 1* базисных дуг образует связующее дерево. ОД решения могут быть получены путем решения связующих деревьев следующим образом:

1. Для дуг, не входящих в связующее дерево (небазисные дуги), соответствующими переменными (*xij or yij) = 0*.
2. Для дуг в связующем дереве (базисные дуги), решим для соответствующих переменных (*xij or yij)* систему линейных уравнений, предоставляемую ограничениями узла.

(Сетевой симплекс метод находит новое ОД решение, начиная с текущего, гораздо более эффективно, без решения этой системы уравнений с нуля). В процессе решения не принимаем во внимание ограничения неотрицательности и ограничения пропускных способностей дуг для базисных переменных  полученное связующее дерево может быть или не

**Допустимое связующее дерево:**

его решение по ограничениям узла удовлетворяет всем другим ограничениям ($0 \le xij \le uij$ *or* $0 \le yij \le uij$).

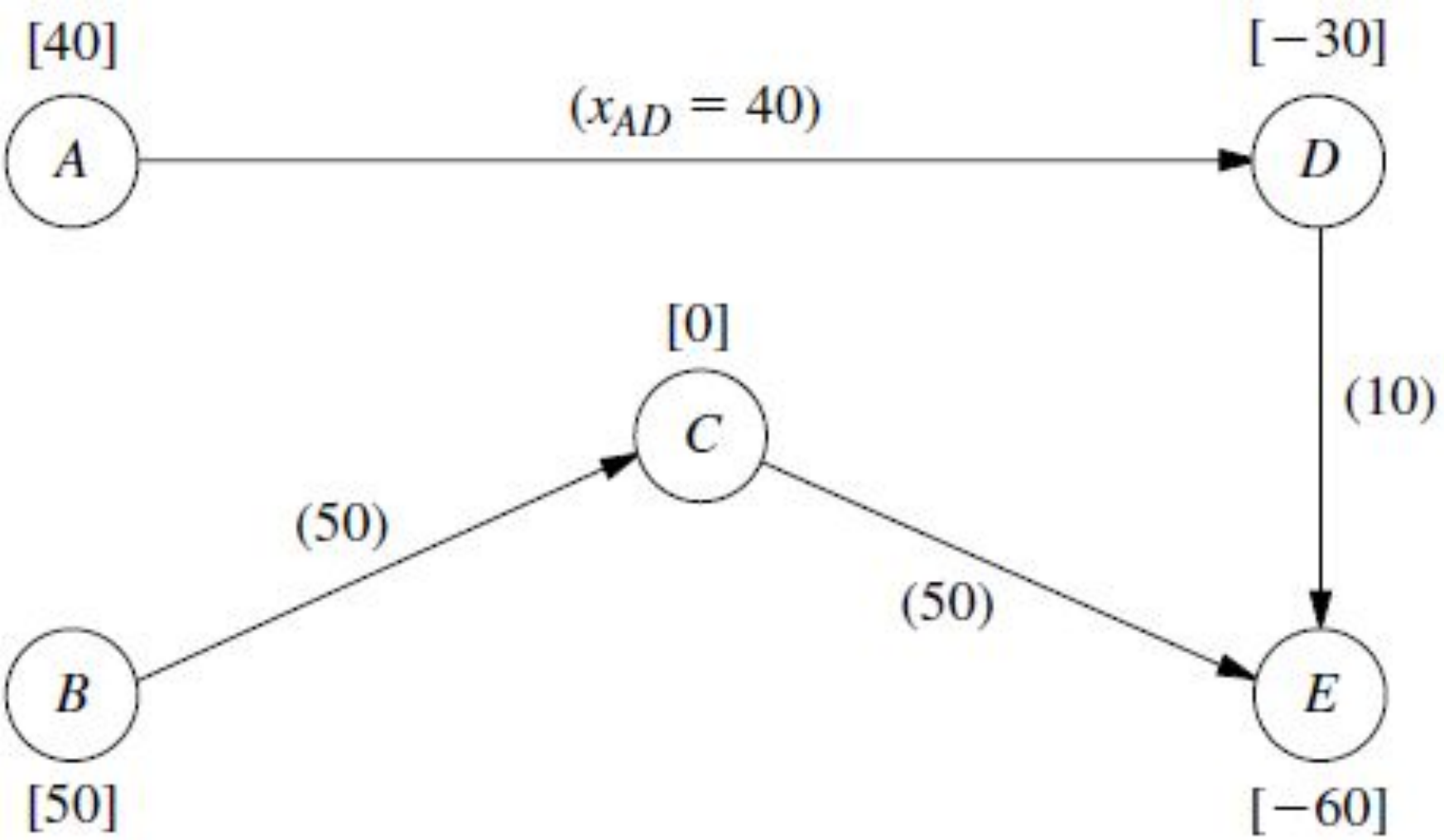
**Фундаментальная теорема сетевого симплекс-метода:**

базисные решения – решения связующего дерева (и наоборот), и эти ОД решения являются решениями для допустимых связующих деревьев (и наоборот). Сеть, получившаяся в результаты замене *xAB = 10 by xAB = 10 - yAB*. Одно связующее дерево для этой сети является то, где дуги *A □ D, D □ E, C □ E, и B □ C.* в качестве базисных дуг. Процесс нахождения решений связующего (остовного) дерева: Слева находятся ограничения узла после замены *xAB* на 10 - *yAB* , где базисные переменные выделены жирным шрифтом. Справа, начиная сверху и двигаясь вниз, шаги для установки или вычисления значений переменных.

$y_{AB} = 0, x_{AC} = 0, x_{ED} = 0$

$$
\begin{aligned}
-y_{AB} + x_{AC} + x_{AD} &= 40 && & x_{AD} = 40.\\
y_{AB} + x_{BC} &= 50 && & x_{BC} = 50.\\
-x_{AC} - x_{BC} + x_{CE} &= 0 && \text{so} & x_{CE} = 50.\\
-x_{AD} + x_{DE} - x_{ED} &= -30 && \text{so} & x_{DE} = 10.\\
-x_{CE} - x_{DE} + x_{ED} &= -60 && \text{Redundant.}
\end{aligned}
$$

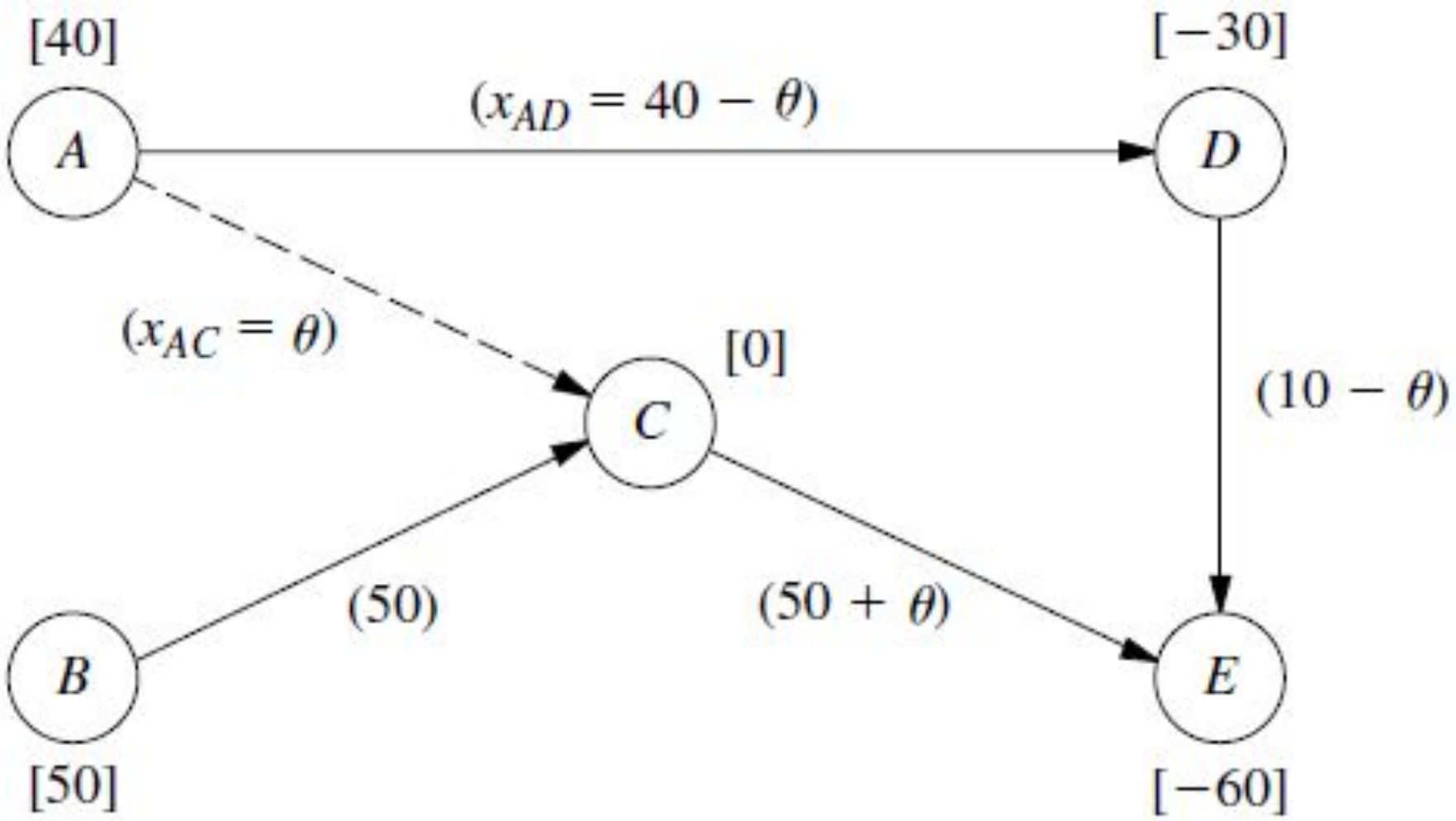
Так как значения базисных переменных удовлетворяют ограничению неотрицательности и одному соответствующему ограничению пропускной способностью дуги (*xCE ≤ 80*), связующее дерево будет допустимым связующим деревом, так что у нас есть ОД решение. Мы используем это решение в качестве исходного ОД решения. Числа данные рядом с дугами представляют *потоки* (значения *xij*), а не едигичную стоимость *cij*. (Скобки вокруг потоков, а не вокруг стоимости).



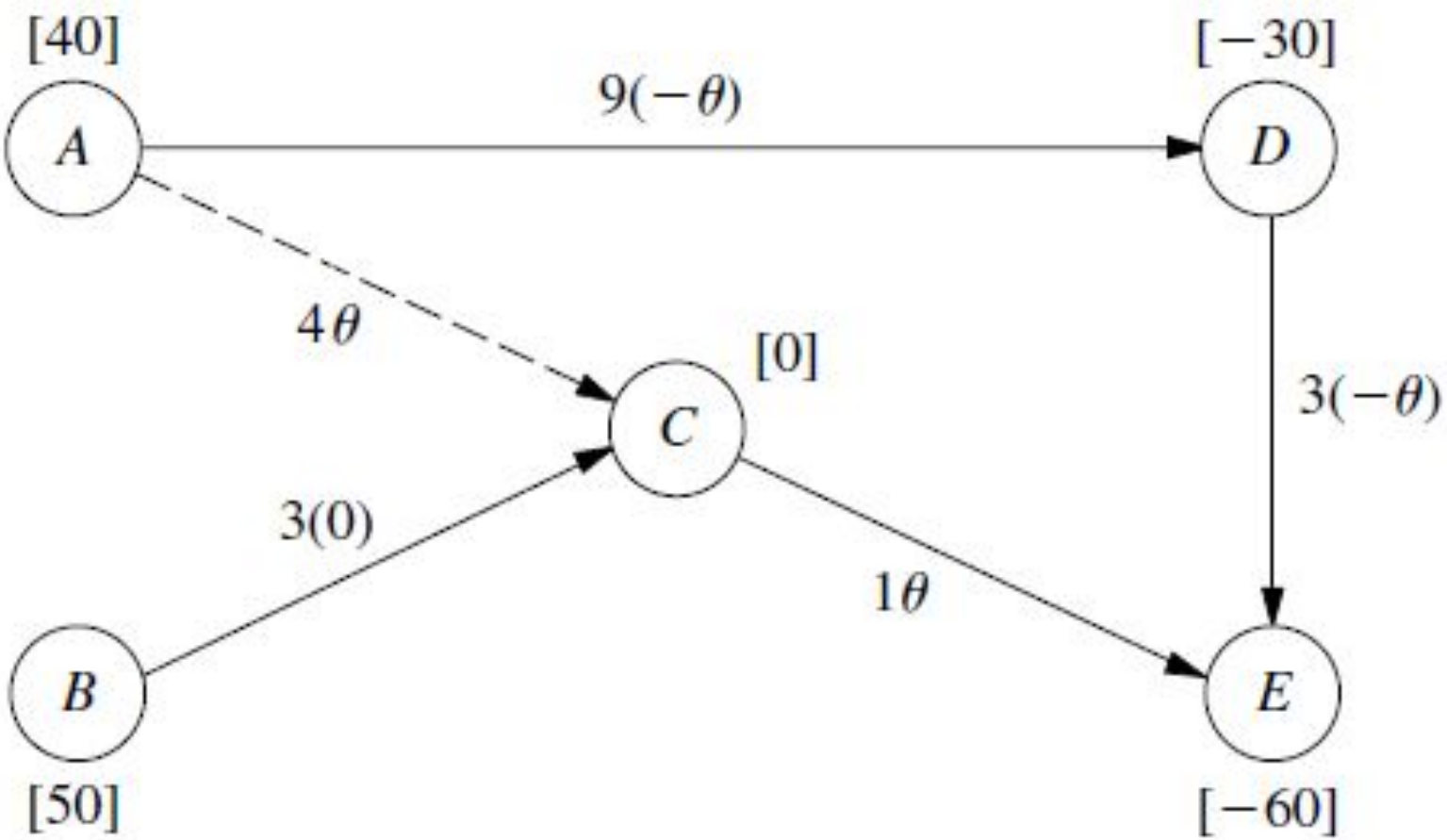

Исходное допустимое связующее дерево и его решение.

**Выбор введенной переменной**

Это выбор небазисной переменной, которая при увеличении с нуля улучшит Z самыми быстрыми темпами. Это делается без симплекс таблицы. Небазисная переменная *xAC* в нашем исходном ОД решении, т. е. не небазисная дуга *A□ C.* Повышение *xAC* от нуля до некоторого значения *θ* означает, что дуга *A□ C* с потоком *θ* должна быть добавлена к сети. Добавление небазисной дуги к связующему дереву всегда создает уникальный неориентированный цикл (*AC–CE–DE–AD*). Поток увеличился на *θ* для других дуг, которые имеют то же направление, что и *A □ C* в цикле (дуга *C□ E*), в то время как поток сети уменьшается на *θ* для других дуг, направление которых противоположно *A□ C* в цикле (дуги *D□ E u A□ D*). Последний, новый поток отменяет поток θ в противоположном направлении. На дуги , которые не в цикле (дуга *B □ C*) новом поток не повлияет. (влияние изменения значения *xAC* на другие переменные в решении получены для начального допустимого связующего дерева).




Влияние на поток добавления дуги *A□ C с потоком θ в исходное допустимое связующее дерево*.




влияние на стоимость добавления дуги *A* □ *C* *c* потоком θ к исходному допустимому связующему дереву

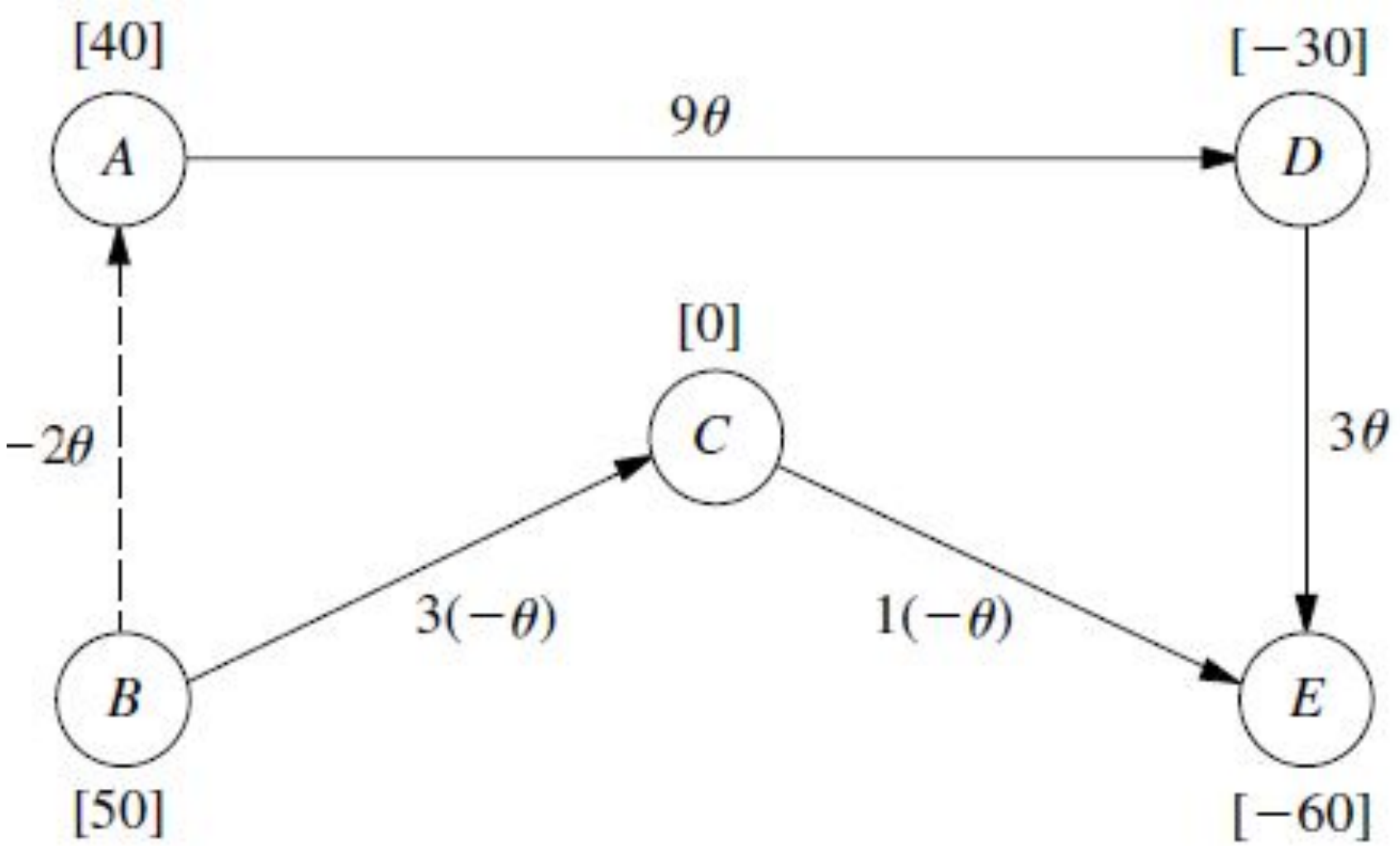
Влияние на Z (общая стоимость потока) от добавления потока $\theta$ к дуге $A \square C$ : единичная стоимость/раз , изменения в потоке для каждой дуги. Общий прирост Z

$\Delta Z = cAC\theta + cCE\theta + cDE(-\theta) + cAD(-\theta) = 4\theta + \theta - 3\theta - 9\theta = -7\theta$.

Присваивание $\theta = 1$ дает скорость изменения Z по мере увеличения $xAC$, $\Delta Z = -7$, когда $\theta = 1$.

Цель: минимизация Z, большая скорость уменьшения Z за счет увеличения $xAC$ очень желательна, поэтому $xAC$ становится главным кандидатом чтобы быть введенной базисной основной переменной.

Прежде чем сделать окончательный выбор вводимых базисных переменных, тот же анализ, проводится для других небазисных переменных. Единственные другие небазисные переменные $yAB$ и $xED$ соответствуют двум другим небазисным дугам $B \square A$ и $E \square D$.




Влияние на стоимость от добавления дуги $B \square A$ с потоком к исходному допустимому связующему дереву.

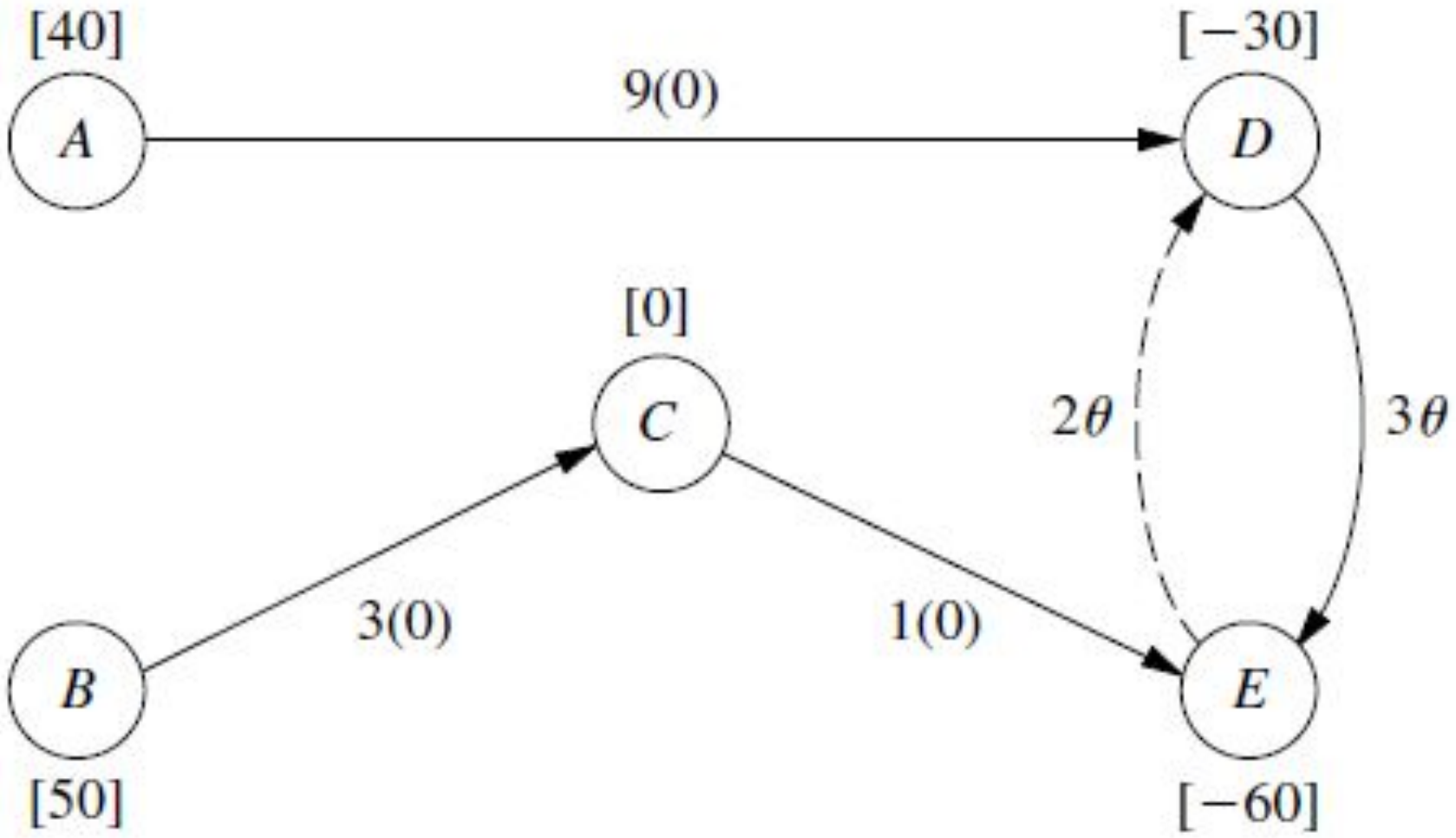
Добавление этой дуги создает неориентированный цикл *BA-AD-DE-CE-BC*, поэтому поток увеличивается на *θ* для дуг *A□ D* и *D□ E* , но уменьшается на *θ* для двух дуг в противоположном направлении, *C□ E uB□ C.* Эти увеличения потока, *θ u-θ,* являются множителями для значений *cij* . *Z = -2θ + 9θ + 3θ + 1(-θ) + 3(-θ) = 6θ* =6, при θ = 1.

Z увеличивается, а не уменьшается, когда *yAB* (поток через обратную дугу B A) увеличивается от нуля, исключает эту переменную в качестве кандидата для вводимой базисной переменной. (Повышение *yAB* от нуля означает уменьшение *xAB*, поток через дугу *A□ B*, от его верхней границы 10). Аналогичный результат получен для последней небазисной дуги *E□ D*. Добавление этой дуги с потоком *θ* к исходному допустимому связующему дереву создает неориентированный цикл *ED-DE*, так что поток увеличивается на θ для дуги *D□ E*, ни на какие другие дуги это больше не влияет.

$\Delta Z = 2\theta + 3\theta = 5\theta = 5$, при $\theta = 1$, поэтому *xED – кандидат на вводимые базисные переменные.*

$$\Delta Z = \begin{cases} -7, & \text{if } \Delta x_{AC} = 1 \\ 6, & \text{if } \Delta y_{AB} = 1 \\ 5, & \text{if } \Delta x_{ED} = 1 \end{cases}$$

Таким образом, отрицательное значение *xAC* означает, что *xAC* становится введенной базисной переменной для первой итерации. Если есть больше, чем одна небазисная переменная с отрицательным значением Z, то выбираем переменную с большим абсолютным значением. (Если нет небазисных переменных с отрицательным значением Z, текущее ОД решение является оптимальным). Вместо выявления неориентированных циклов и т.д., сетевой симплекс методом получает эти значения Z с помощью алгебраической процедуры, и является более эффективным (особенно для больших сетей).




Влияние на стоимость добавления дуги $E \square D$ с потоком $\theta$ в исходное допустимое связующее дерево.

**Нахождение уходящей базисной переменной и следующее ОД решение**

После выбора вводимой базисной переменной определяется уходящая базисная переменная и следующее ОД решение.

**Итерация 1**: так как *xAC* является вводимой базисной переменной, поток θ через дугу *A C* должен быть увеличен с нуля, насколько это возможно, пока одна из базисных переменных достигает своей нижней грани (0) или верхней границы (*uij*) . Для тех дуг, поток которых увеличивается с *θ* (дуги *A C* и *C E*), необходимо учитывать только верхние границы *(uAC =∞ and uCE = 80 ) :*

$$x_{AC} = \theta \leq \infty.$$
$$x_{CE} = 50 + \theta \leq 80, \qquad \text{so} \qquad \theta \leq 30.$$

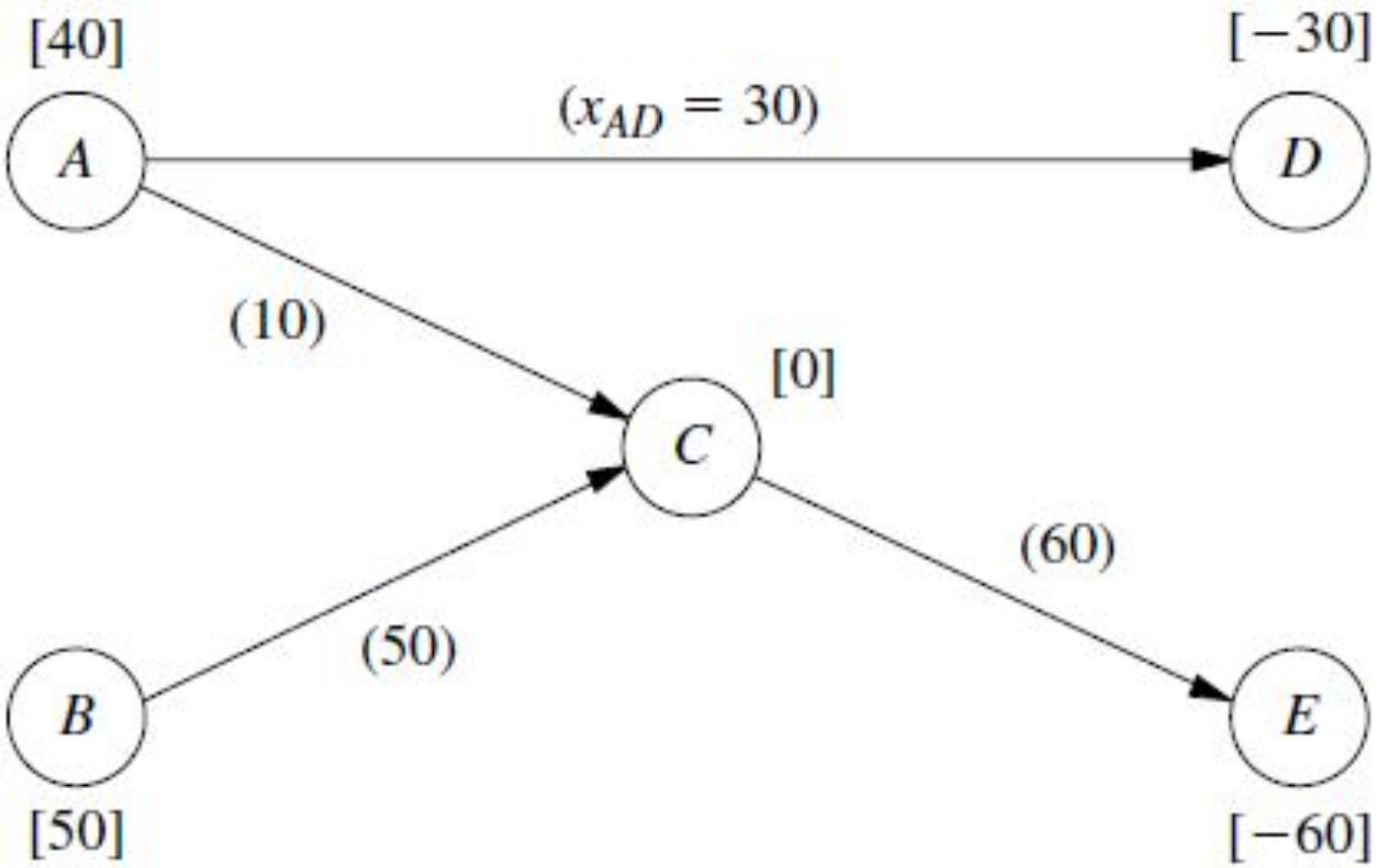
Для тех дуг, поток которых уменьшается с *θ (дуги D E и A D), необходимо учитывать только нижнюю границу* 0 :

$$x_{DE} = 10 - \theta \geq 0, \qquad \text{so} \qquad \theta \leq 10.$$
$$x_{AD} = 40 - \theta \geq 0, \qquad \text{so} \qquad \theta \leq 40.$$

Дуги, поток которых не изменяется на θ (не входят в неориентированный цикл), - только дуга $B\square C$, могут быть проигнорированы, так как по мере увеличения $\theta$ граница не будет достигнута.
Для пяти дуг, $xDE$ должна быть уходящей базисной переменной, поскольку она достигает границы наименьшего значения θ (10). Задаем θ = 10 , это дает поток через базисные дуги в следующем ОД решении:

$$
\begin{aligned}
x_{AC} &= \theta = 10, \\
x_{CE} &= 50 + \theta = 60, \\
x_{AD} &= 40 - \theta = 30, \\
x_{BC} &= 50.
\end{aligned}
$$

Если уходящая базисная переменная достигла верхней границы, то нужна корректировка метода верхней грани. Так как была достигнута нижняя граница 0, больше ничего не нужно делать.




Второе допустимое связующее дерево и его решение.

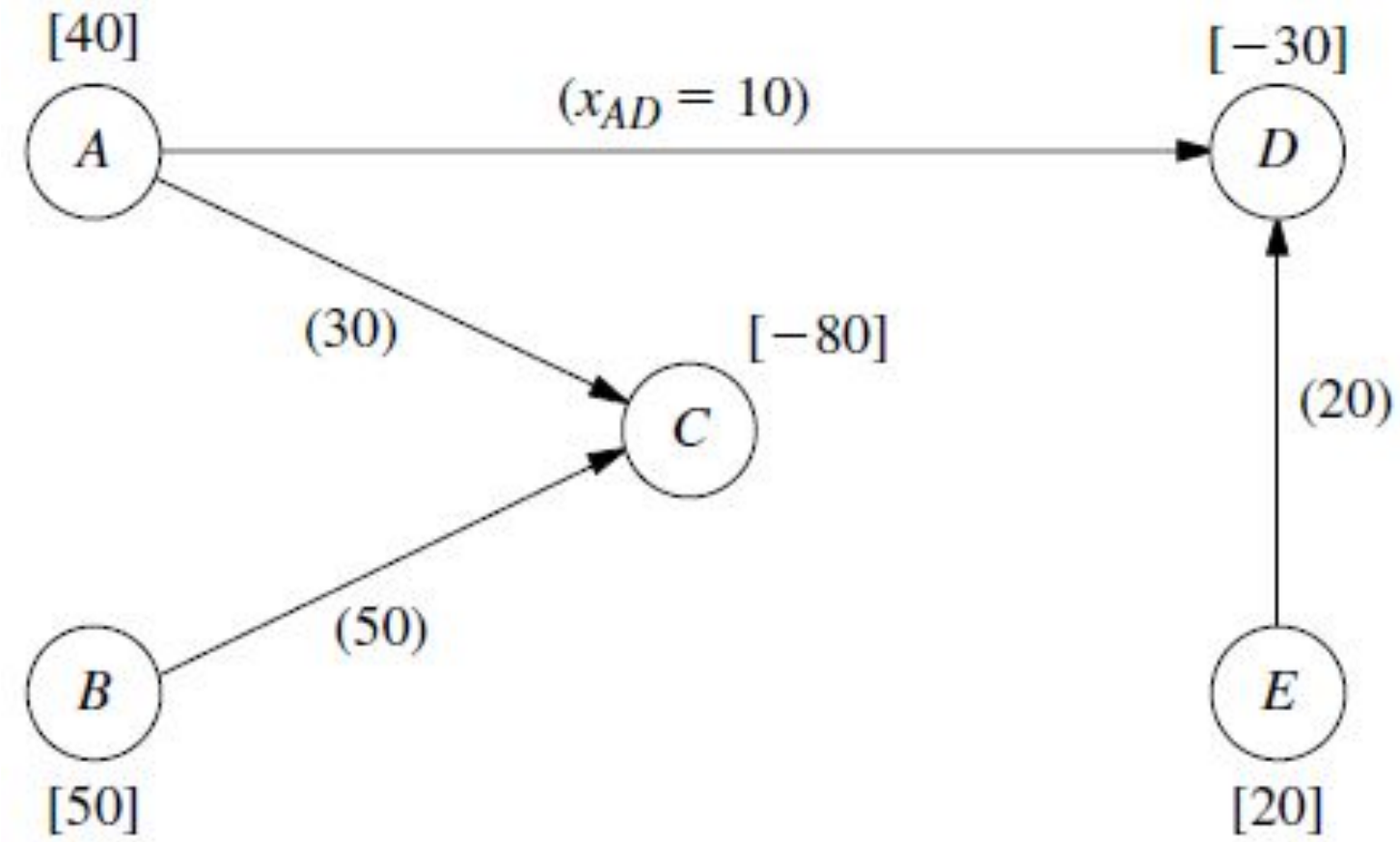
| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ | |
|---|---|---|---|
| $B \rightarrow A$ | $BA–AC–BC$ | $-2 + 4 - 3 = -1$ | |
| $D \rightarrow E$ | $DE–CE–AC–AD$ | $3 - 1 - 4 + 9 = 7$ | |
| $E \rightarrow D$ | $ED–AD–AC–CE$ | $2 - 9 + 4 + 1 = -2$ | ← Minimum |

Вычисления для выбора вводимой базисной переменной в Итерации 2

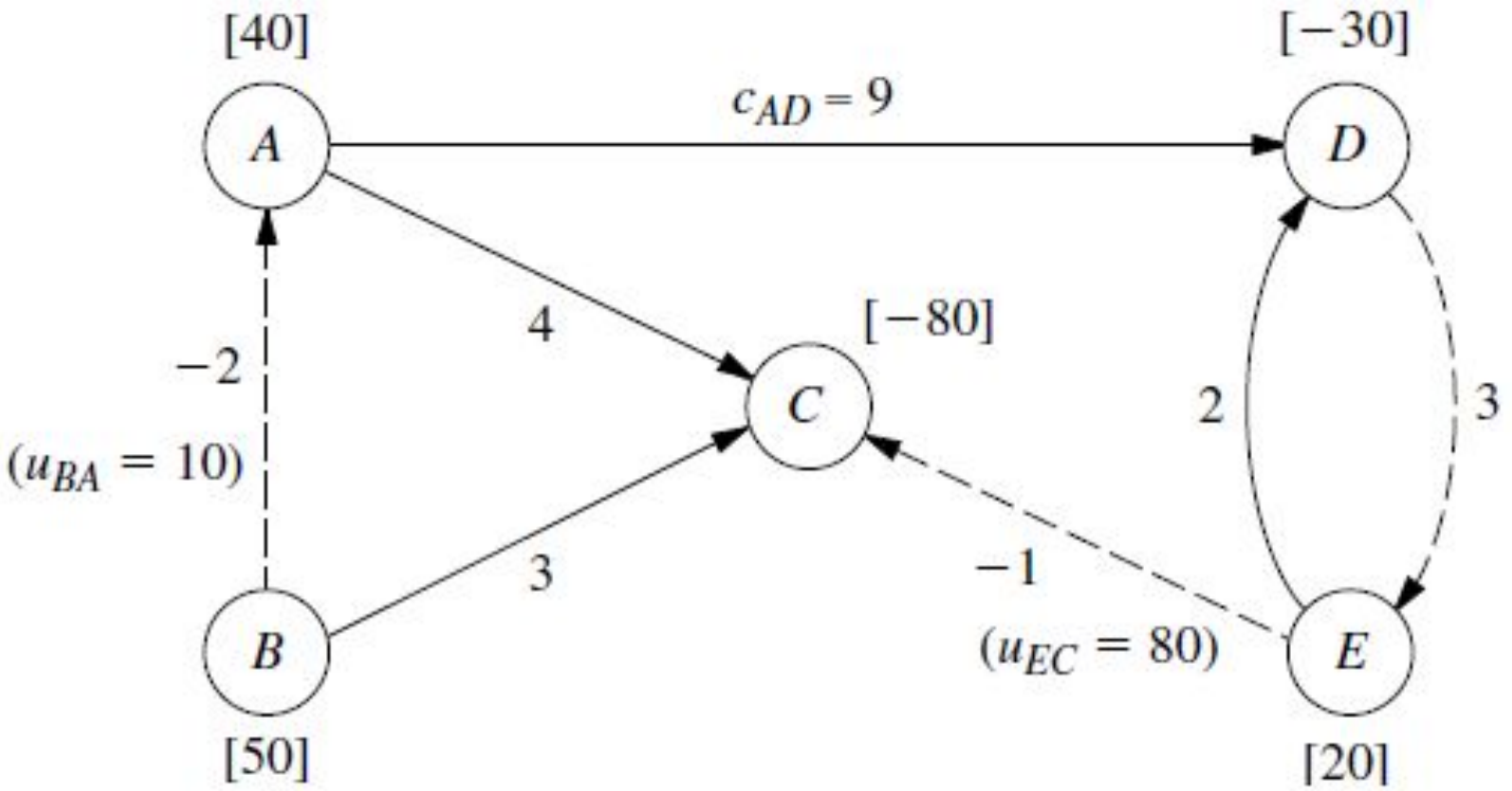
**Итерация 2:** Начиная с допустимого связующего дерева и учитывая единичную стоимость *cij*, мы переходим к расчетам для выбора вводимой базисной переменной. Во второй колонке таблицы указан уникальный неориентированный цикл, который создается путем добавления небазисных дуг в первой колонке для этого связующего дерева, и 3-я колонка показывает влияние на затраты, вызванные изменениями в потоках в этом цикле из-за добавления потока $\theta = 1$ к неосновной дуге. Дуга *ED* имеет наибольшее отрицательное значение $\Delta Z$, поэтому *xED* – вводимая базисная переменная. Поток θ через арку E □ D сделан как можно большим, в то же время удовлетворяя следующим границам потока:

$$x_{ED} = \theta \le u_{ED} = \infty, \quad \text{so} \quad \theta \le \infty.$$
$$x_{AD} = 30 - \theta \ge 0, \quad \text{so} \quad \theta \le 30.$$
$$x_{AC} = 10 + \theta \le u_{AC} = \infty, \quad \text{so} \quad \theta \le \infty.$$
$$x_{CE} = 60 + \theta \le u_{CE} = 80, \quad \text{so} \quad \theta \le 20. \quad \leftarrow \text{Minimum}$$

Так как *xCE* накладывает наименьшие верхние границы (20) на *θ*, *xCE* становится уходящей базисной переменной. Назначение θ = 20 в приведенных выше выражениях для *xED, xAD, uxAC* □ поток через базисные дуги для следующего ОД решения (при *xBC = 50* не зависит от *θ*).




Третье допустимое связующее дерево и его решение.




Скорректированная сеть с единичными стоимостями в конце итерации 2.

Уходящая базисная переменная *xCE* полученная достижением верхней границы (80). Используя метод верхней границы, *xCE* заменена на *80 - yCE*, где *yCE* = 0 - новая небазисная переменная. Исходная дуга *C□ E* с *cCE = 1 and uCE = 80* заменяется на обратную дугу *E□C* с *cEC1 и uEC = 80* . Значения *bE и bC* также корректируется путем прибавления 80 к *bE* и вычитания 80 из *bC*. Получившаяся скорректированная сеть, небазисные дуги показаны пунктирными линиями и числа около всех дуг единичная стоимость.

**Итерация 3:** расчеты приводят к выбору *yAB* (обратная дуга *B□ A*) в качестве вводимой базисной переменной, как показано в таблице. Затем добавить столько потка через дугу *B□ A* сколько это возможно, удовлетворяя границам потока ниже:

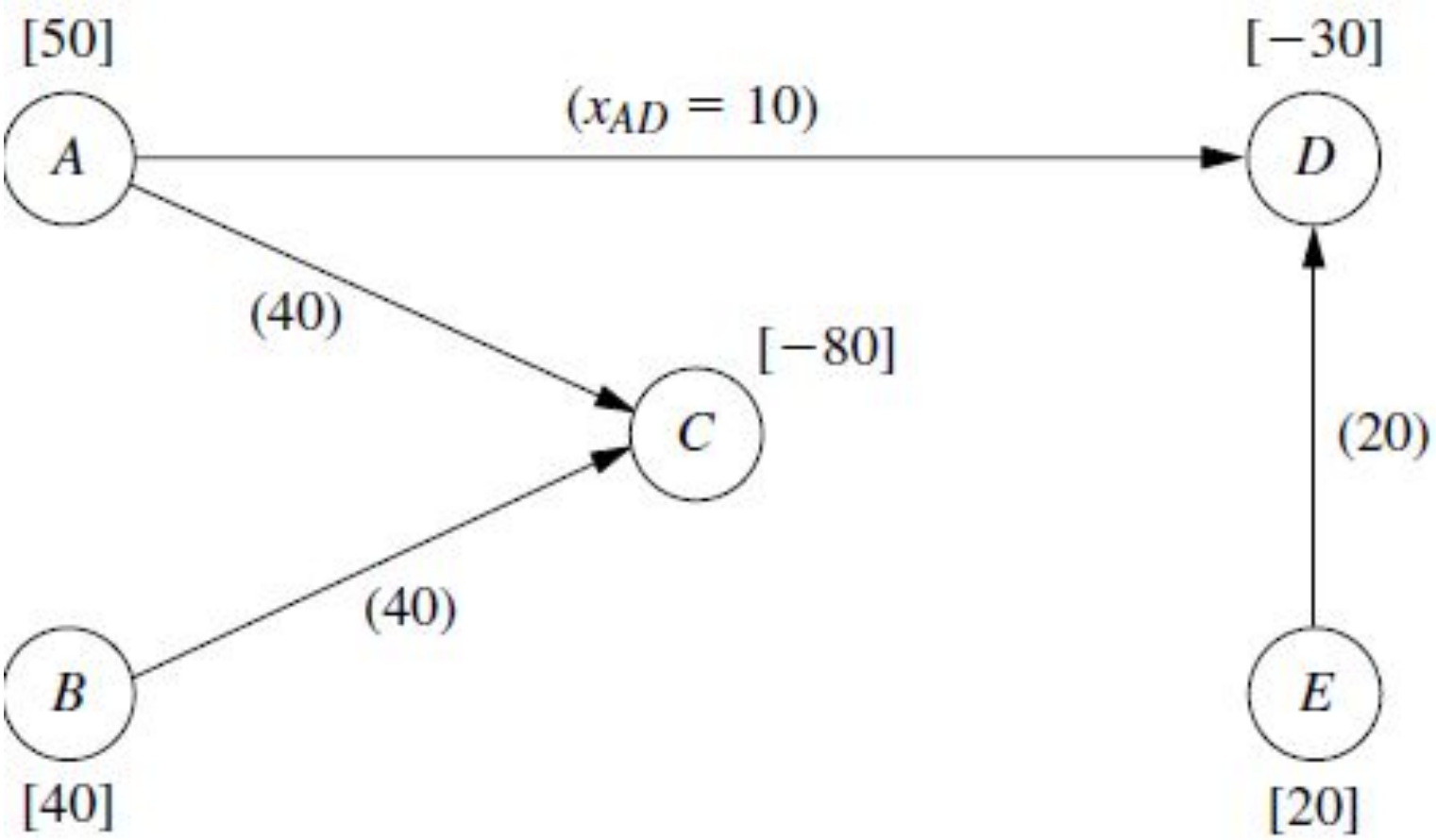
$$y_{AB} = \theta \le u_{BA} = 10, \quad \text{so} \quad \theta \le 10. \quad \leftarrow \text{Minimum}$$
$$x_{AC} = 30 + \theta \le u_{AC} = \infty, \quad \text{so} \quad \theta \le \infty.$$
$$x_{BC} = 50 - \theta \ge 0, \quad \text{so} \quad \theta \le 50.$$

Наименьшая верхняя граница (10) на θ накладывается *yAB*, так что эта переменная становится уходящей базисной переменной. Устанавливая θ = 10 в этих выражений для *xAC и xBC* (наряду с неизменными значениями *xAC = 10 и xED = 20*), получим ОД следующее решение. Как и 2-го, оставив основной переменной (ЯБ), полученные здесь переменная достигнув верхней границы.Ввод основных переменных ЯБ также стал оставляя основную переменную на той же итерации!

Smallest upper bound (10) on θ is imposed by *yAB, so this variable becomes the leaving* basic variable. Setting θ = 10 in these expressions for *xAC and xBC (along with* unchanged values of *xAC = 10 and xED = 20) then yields the next BF solution.* As with iteration 2, the leaving basic variable (*yAB) obtained here by the variable* reaching its upper bound. The *entering basic*

| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ | |
|---|---|---|---|
| $B \rightarrow A$ | $BA–AC–BC$ | $-2 + 4 - 3 = -1$ | ← Minimum |
| $D \rightarrow E$ | $DE–ED$ | $3 + 2 = 5$ | |
| $E \rightarrow C$ | $EC–AC–AD–ED$ | $-1 - 4 + 9 - 2 = 2$ | |

Calculations for selecting the entering basic variable for iteration 3



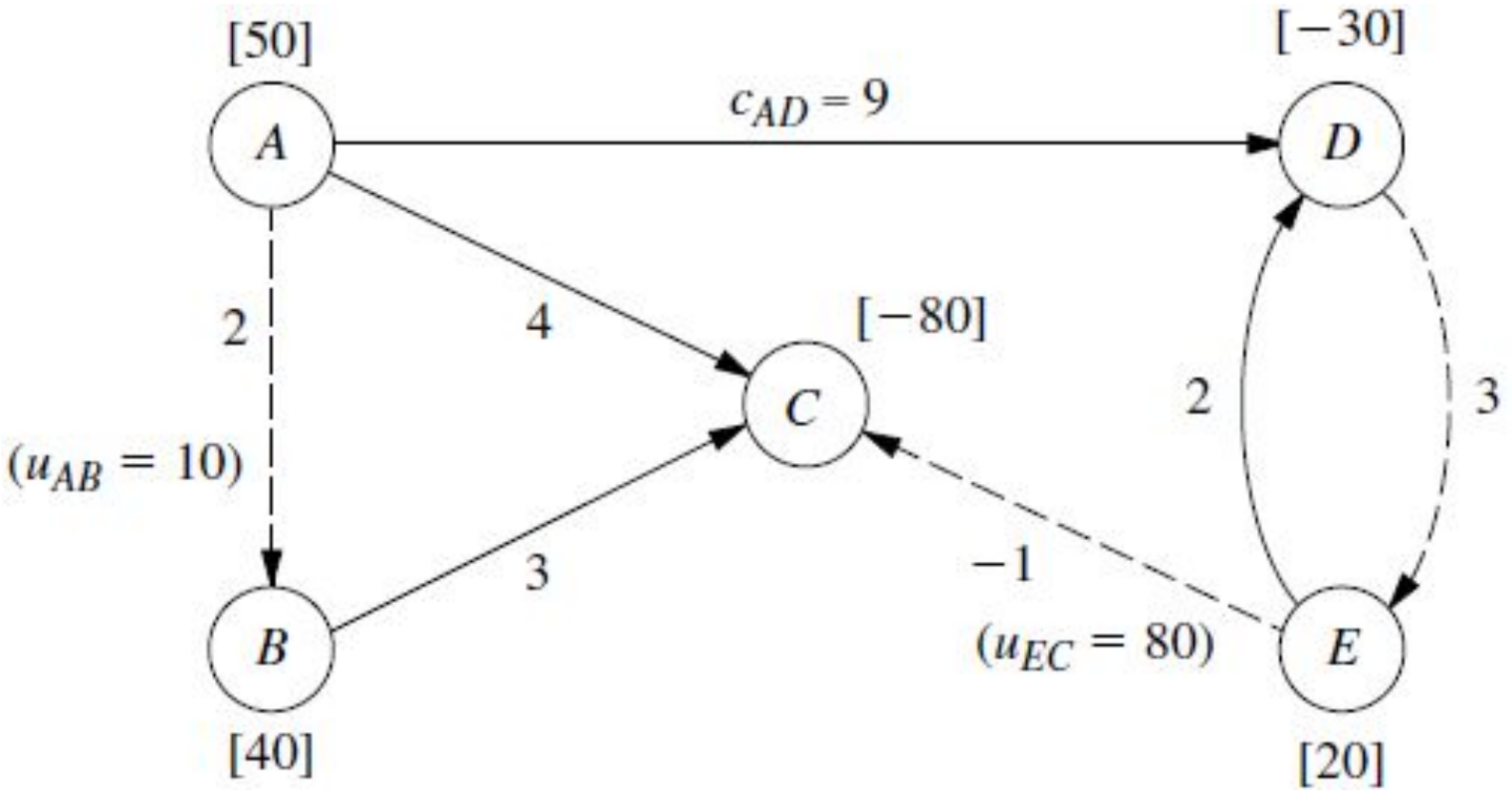

fourth (and final) feasible spanning tree and its solution.

Increasing the entering basic variable from zero causes *its upper bound to be reached first before any other basic variables reach a bound. A*rc *B*□ *A now needs to be replaced by a reverse arc A*□ *B (because of the leaving basic variable reaching an upper bound) already* is a reverse arc! The reverse arc for a reverse arc is simply the original *real arc. Therefore, the arc B*□ *A (with cBA = 2 and uBA = 10)* is replaced by arc *A*□ *B (with cAB = -2 and uAB = 10), which is the arc between* nodes *A and B in original network, and a generated net flow* of 10 is shifted from node *B (bB=50*□ *40) to node A (bA 40*□ *50). V*ariable *yAB =10 is replaced by 10 - xAB, with xAB = 0 as the new nonbasic variable.*

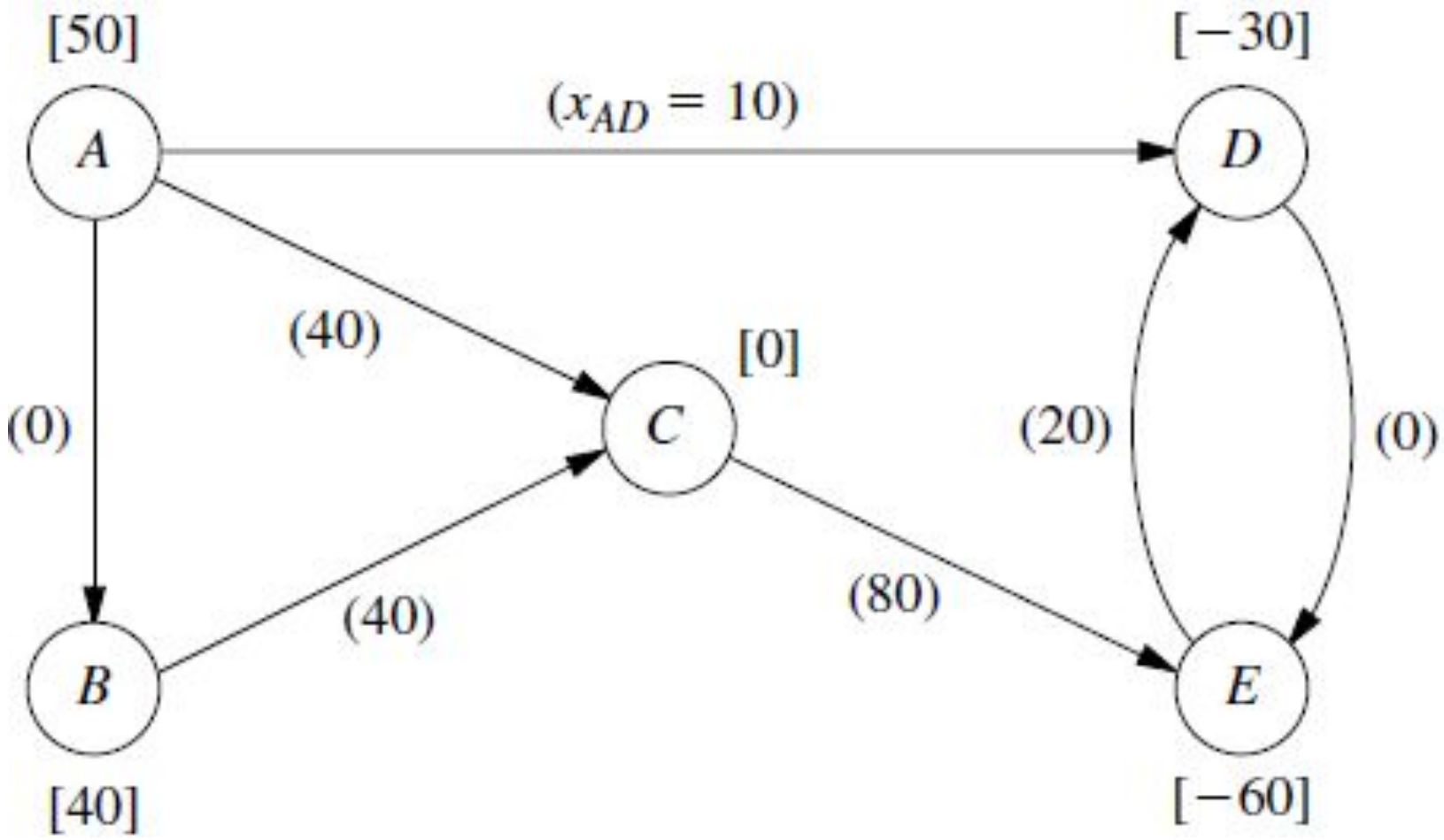
***Passing the Optimality Test:*** *Algorithm will* find the next entering basic variable with the usual calculations. However, *none of the nonbasic arcs gives a negative value of Z, so an improvement* in *Z cannot be achieved by introducing flow through any of them.*

| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ |
|---|---|---|
| $A \rightarrow B$ | $AB–BC–AC$ | $2 + 3 - 4 = 1$ |
| $D \rightarrow E$ | $DE–EC–AC–AD$ | $3 - 1 - 4 + 9 = 7$ |
| $E \rightarrow C$ | $EC–AC–AD–ED$ | $-1 - 4 + 9 - 2 = 2$ |

Calculations for the optimality test at the end of iteration 3




adjusted network with unit costs at the completion of iteration 3.




Optimal flow pattern in original network for the Distribution Co. example.

*This means* that the current BF solution *passed the optimality test, so the algorithm* stops. To identify flows through real arcs rather than reverse arcs for this optimal solution, the current adjusted network should be compared with the original network. Each arc has same direction in the two networks with the one exception of the arc between nodes *C and E, i.e. the only re*verse arc is arc *E□ C, where its flow is given by variable yCE.*

*Therefore,* calculate *xCE = uCE- yCE = 80 - yCE.*

*Arc E□ C happens to be a non-basic arc,* so *yCE = 0 and xCE = 80 is the flow through the real arc C□ E.*